This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problems Mailbox.

This Page Blank (uspto)

R E P U B L I Q U E F R A N Ç A I S

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

FR 00/625
4

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

REC'D 10 APR 2000
WIPO PCT

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 27 MARS 2000

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

M Haut

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS CONFORMÉMENT À LA REGLE 17.1.a) OU b)

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS Cédex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30

This Page Blank (uspto)

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

BREVET D'INVENTION, CERTIFICAT D'UTILITE

Code de la propriété intellectuelle-Livre VI

REQUÊTE EN DÉLIVRANCE

Cerfa N° 55 -1328

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

Réservé à l'INPI

DATE DE REMISE DES PIÈCES 15 MARS 1999

N° D'ENREGISTREMENT NATIONAL 99 03172

DÉPARTEMENT DE DÉPÔT 75

DATE DE DÉPÔT 15 MARS 1999

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

BULL S.A.
Monsieur Bernard CORLU / PC 58F35
68, route de Versailles
78434 LOUVECIENNES CEDEX

n°du pouvoir permanent PG 4280 | références du correspondant FR3815/BC | téléphone 01 39.66.61.76

2 DEMANDE Nature du titre de propriété industrielle

☒ brevet d'invention ☐ demande divisionnaire ☐ certificat d'utilité ☐ transformation d'une demande de brevet européen

demande initiale : ☐ brevet d'invention ☐ certificat d'utilité n° date

Établissement du rapport de recherche ☐ différé ☒ immédiat

Le demandeur, personne physique, requiert le paiement échelonné de la redevance ☐ oui ☒ non

Titre de l'invention (200 caractères maximum)

Procédé d'accès à un objet à l'aide d'un navigateur de type "web" coopérant avec une carte à puce et architecture pour la mise en œuvre du procédé.

3 DEMANDEUR (S) n° SIREN 3 2 9 5 5 6 1 4 6 code APE-NAF B 3 2 1

Nom et prénoms (souligner le nom patronymique) ou dénomination

BULL CP8

Forme juridique

S.A.

Nationalité (s) Française

Adresse (s) complète (s) Pays

BULL CP8
BP 45
68, route de Versailles
78430 LOUVECIENNES

FRANCE

En cas d'insuffisance de place, poursuivre sur papier libre ☐

4 INVENTEUR (S) Les inventeurs sont les demandeurs ☐ oui ☒ non Si la réponse est non, fournir une désignation séparée

5 RÉDUCTION DU TAUX DES REDEVANCES ☐ requise pour la 1ère fois ☐ requise antérieurement au dépôt : joindre copie de la décision d'admission

6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE

pays d'origine | numéro | date de dépôt | nature de la demande

7 DIVISIONS antérieures à la présente demande n° date n° date

8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (nom et qualité du signataire)

Bernard CORLU
Mandataire -

SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION SIGNATURE APRÈS ENREGISTREMENT DE LA DEMANDE À L'INPI

La loi n°78-17 du 6 janvier 1978 relative à l'informatique aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

DEPARTEMENT DES BREVETS

26bis, rue de Saint-Pétersbourg
75800 Paris Cédex 08
Tél. : 01 53 04 53 04 - Télécopie : 01 42 93 59 30

BREVET D'INVENTION CERTIFICAT D'UTILITE

FR 3815/BC

DÉSIGNATION DE L'INVENTEUR
(si le demandeur n'est pas l'inventeur ou l'unique inventeur)

N° D'ENREGISTREMENT NATIONAL

99 03172 du 15 mars 1999

TITRE DE L'INVENTION :

"PROCEDE D'ACCES A UN OBJET A L'AIDE D'UN NAVIGATEUR DE TYPE "WEB" COOPERANT AVEC UNE CARTE A PUCE ET ARCHITECTURE POUR LA MISE EN ŒUVRE DU PROCEDE."

LE(S) SOUSSIGNÉ(S)

BULL S.A.

DÉSIGNE(NT) EN TANT QU'INVENTEUR(S) (indiquer nom, prénoms, adresse et souligner le nom patronymique)

Urien Pascal
4 rue du Ruisseau St Prix
78450 VILLEPREUX
France

NOTA : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur (s) ou du mandataire

Louveciennes, le 19 Mars 1999

Corlu Bernard (mandataire)

RA 113/140897

PROCEDE D'ACCES A UN OBJET A L'AIDE D'UN NAVIGATEUR DE TYPE "WEB" COOPERANT AVEC UNE CARTE A PUCE ET ARCHITECTURE POUR LA MISE EN ŒUVRE DU PROCEDE.

L'invention concerne un procédé d'accès à un objet à l'aide d'un navigateur de type "WEB" coopérant avec une carte à puce, plus particulièrement d'accès à un objet qui sera appelé virtuel ci-après.

L'invention concerne plus particulièrement encore un procédé d'accès sécurisé aux objets précités.

L'invention concerne également une architecture de mise en œuvre d'un tel procédé.

Le navigateur "WEB" est compris dans une station d'utilisateur. Cette station d'utilisateur est connectée à un réseau de transmission de données, et notamment de type Internet.

Dans le cadre de l'invention, le terme "objet" doit être considéré dans son sens le plus général. Il englobe de nombreux types de ressources informatiques, tels que des fichiers textes, des fichiers images ou des fichiers multimédias (vidéo, son, etc.). Il englobe également des transactions ou des connexions à un système informatique, selon un protocole donné.

Dans le premier cas, on parlera ci-après d'objets statiques, car leur instance ne dépend pas du temps. Dans le second cas, on parlera d'objets dynamiques, car leur instance varie en fonction du temps. On peut citer, à titre d'exemple non limitatif, dans le cadre d'un réseau de type Internet, une connexion de type "Telnet".

Toujours dans le cadre de l'invention, le terme "station d'utilisateur" doit être compris dans un sens général. La station d'utilisateur précitée peut être notamment constituée par un ordinateur personnel fonctionnant sous divers systèmes d'exploitation, tels WINDOWS ou UNIX (tous deux étant des marques déposées). Elle peut être aussi constituée par une station de travail, un ordinateur portable ou un terminal de carte, dit dédié.

De même, dans le cadre de l'invention, le terme "réseau Internet" englobe, outre le réseau Internet proprement dit, les réseaux privés d'entreprises ou similaires, du type dit "intranet", et les réseaux les prolongeant vers l'extérieur, du type dit "extranet", de façon générale tout réseau dans lequel les échanges de données s'effectuent selon un protocole du type Internet.

Dans ce qui suit, sans en limiter en quoi que ce soit la portée, on se placera dans le cadre de l'application préférée de l'invention, sauf mention contraire. On considérera donc une station d'utilisateur, que l'on appellera simplement "terminal", munie d'un lecteur de carte à puce et connecté à un réseau de type Internet.

Un système d'application à base de carte à puce comporte généralement les éléments principaux suivants :

- une carte à puce ;
- un système hôte constituant le terminal précité ;
- un réseau de communication, à savoir le réseau Internet dans l'application préférée ;
- et un serveur d'application connecté au réseau.

La figure 1A illustre schématiquement un exemple d'architecture de ce type. Le terminal 1, par exemple un ordinateur individuel, comporte un lecteur 3 de carte à puce 2. Ce lecteur 3 peut être ou non physiquement intégré dans le terminal 1. La carte à puce 2 comporte un circuit intégré 20 dont des connexions d'entrées-sorties affleurent en surface de son support pour autoriser une alimentation en énergie électrique et des communications avec le terminal 1. Ce dernier comprend des circuits d'accès à un réseau de transmissions de données *RI*. Ces circuits dépendent, notamment, de la nature du réseau *RI* et du terminal 1. A titre d'exemple, il peut s'agir d'une carte réseau pour un réseau de type local ou d'un modem pour se connecter à une ligne téléphonique commutée ou à un réseau numérique à intégration de services ("RNIS"), pour se connecter au réseau Internet, par exemple via un prestataire de services Internet ("Internet Service Provider" ou "ISP", selon la terminologie anglo-saxonne).

Le terminal 1 comprend naturellement tous les circuits et organes nécessaires à son bon fonctionnement, et qui n'ont pas été représentés dans un but de simplification du dessin : unité centrale, mémoires vive et fixe, mémoire de masse à disque magnétique, lecteur de disquette et/ou de CédéRom, etc.

Habituellement, le terminal 1 est aussi relié à des périphériques classiques, intégrés ou non, tels un écran de visualisation 5 et un clavier 6.

Le terminal 1 peut être mis en communication avec des serveurs ou tous systèmes informatiques connectés au réseau *RI*, dont un seul, 4, est illustré sur la figure 1A. Dans le cas de l'application préférée de l'invention, les circuits d'accès 11 mettent le terminal 1 en communication avec les serveurs 4 grâce à un logiciel particulier 10, appelé navigateur de type "WEB", ou "browser" selon la terminologie anglo-saxonne. Celui-ci permet d'accéder à diverses applications réparties sur l'ensemble du réseau *RI*, généralement selon un mode "client-serveur".

Habituellement, les communications sur les réseaux s'effectuent conformément à des protocoles répondant à des standards comprenant plusieurs couches logicielles superposées. Dans le cas d'un réseau *RI* de type Internet, les communications s'effectuent selon des protocoles spécifiques à ce type de communications, qui seront détaillés ci-après, mais qui comprennent également plusieurs couches logicielles. Le protocole de communication est choisi en fonction de l'application plus particulièrement visée : interrogation de pages "WEB", transferts de fichiers, courrier électronique (e-mel, ou "e-mail" selon la terminologie anglo-saxonne), forums ou nouvelles ("news" selon la terminologie anglo-saxonne), etc.

L'architecture logique du système comprenant un terminal, un lecteur de carte à puce et la carte à puce, est représentée schématiquement par la figure 1B. Elle est décrite par la norme ISO 7816, qui elle-même comportent plusieurs sous-ensembles :

- ISO 7816-1 et 7816-2, en ce qui concerne les dimensions et le marquage des cartes ;

- ISO 7816-3, en ce qui concerne le transfert de données entre le terminal et la carte à puce ; et
- ISO 7816-4, en ce qui concerne la structure du jeu d'ordres et le format des commandes.

Sur la figure 1B, du côté terminal 1, on n'a représenté que les couches répondant à la norme ISO 7816-3, référencées 101, et le gestionnaire d'ordres "APDU" (norme ISO 7816-4), référencé 102. Du côté carte à puce 2, les couches répondant à la norme ISO 7816-3 sont référencées 200 et le gestionnaire d'ordres "ADPU" (norme ISO 7816-4) est référencé 201. Les applications sont référencées A_1, ..., A_i, ..., A_n ; n étant le nombre maximum d'applications présentes sur la carte à puce 2.

Une application "cardlet", A_i, présente dans la carte à puce 2 (figure 1A), dialogue avec le terminal 1 au moyen d'un jeu d'ordres. Ce jeu présente typiquement des ordres d'écriture et des ordres de lecture. Le format des ordres est connu sous l'abréviation anglo-saxonne de "APDU" (pour "Application Protocol Data Unit"). Il est défini par la norme ISO 7816-4 précitée. Un "APDU" de commande est noté "*APDU.command*" et un "APDU" de réponse est noté "*APDU.response*". Les "APDU" sont échangés entre le lecteur de carte et la carte à puce au moyen d'un protocole spécifié par la norme ISO 7816-3 précitée (par exemple en mode caractère : T=0 ; ou en mode bloc : T=1).

Lorsque la carte à puce 2 inclut plusieurs applications distinctes, comme illustré sur la figure 1B, on parle de carte multi-applicative. Cependant, le terminal 1 dialogue avec une seule application à la fois. Une application A_i se présente, par exemple, sous la forme d'une pièce de logicielle, dite "applet", en langage "JAVA" (marque déposée), que l'on appellera ci-après "cardlet". La sélection d'un "cardlet" particulier A_i est obtenu à l'aide d'un "APDU" du type sélection ("SELECT"). Une fois ce choix effectué, les "APDU" qui le suivent sont acheminés vers ce "cardlet". Un "APDU SELECT" nouveau a pour effet d'abandonner l'application en cours et d'en choisir une autre. Le sous-ensemble logiciel gestionnaire des "APDU" 201 permet de choisir une

application particulière A_i dans la carte à puce 2, de mémoriser l'application ainsi choisie, et de transmettre et/ou recevoir des "APDU" vers et depuis cette application.

En résumé de ce qui vient d'être décrit, la sélection d'une application A_i et le dialogue avec celle-ci s'effectue par échanges d'ordres "APDU". On suppose que les applications A_i sont des applications conventionnelles, que l'on appellera ci-après "GCA" (pour "Generic Card Application" ou application de carte générique).

Dans un système d'applications à base de carte à puce, comme illustré par l'architecture de la figure 1B, cette dernière peut se voir dévolu diverses fonctions, notamment des fonctions de sécurité. Il est en effet avantageux de stocker les données liées à la sécurité (mots de passe, droits d'accès, etc.) dans une carte à puce qui peut être conservée par l'utilisateur. En outre les données étant enregistrées dans une mémoire fixe, sous une forme qui peut être chiffrée, elles ne sont pas facilement modifiables, ni même directement lisibles de l'extérieur.

Cependant, il est à noter que la carte 3 ne peut communiquer directement avec les navigateurs du commerce, sauf à modifier le code de ces derniers. Les cartes à puce actuelles, qui par ailleurs sont conformes aux standards rappelés ci-dessus, ont une configuration matérielle et logicielle qui ne permet pas non plus de communiquer directement avec le réseau Internet. En particulier, elles ne peuvent recevoir et transmettre des paquets de données, selon l'un ou l'autre des protocoles utilisés sur ce type de réseau. Il est donc nécessaire de prévoir une pièce de logiciel additionnelle, implantée dans le terminal 1, généralement sous la forme de ce qui est appelé un "plug-in", selon la terminologie anglo-saxonne. Cette pièce de logiciel, qui porte la référence 12 sur la figure 1A, effectue l'interface entre le navigateur 10 et la carte 2, plus précisément les circuits électroniques 20 de cette carte 2.

Dans l'état actuel de la technique, le système hôte associé au lecteur de carte 3, c'est-à-dire le terminal 1, est associé également à une application

particulière. En d'autres termes, il est nécessaire de prévoir un terminal spécifique, dit "dédié", pour chaque application particulière.

En outre, il est clair que, même compte tenu de la rapide évolution passée des technologies et de leur évolution future prévisible, la capacité d'enregistrement d'informations dans des circuits de mémoire, vive ou fixe, d'une carte à puce reste et restera très limitée, si on compare cette capacité à celle offerte par un terminal "hôte" de cette carte à puce, et naturellement à celles offertes par des systèmes plus importants, "mini-ordinateurs" ou grands systèmes de type dit "main frame". Aussi, il n'est pas possible de stocker les données d'un nombre important d'applications dans une carte à puce, et notamment des fichiers de type multimédia très volumineux.

L'invention vise à pallier les inconvénients des dispositifs de l'art connu et dont certains viennent d'être rappelés, tout en répondant aux besoins qui se font sentir. Il doit notamment être possible d'accéder à un grand nombre d'applications, même volumineuses d'un point de vue quantité de données, de natures diverses et réparties sur tout le réseau Internet. En outre, dans un mode de réalisation préféré, les accès doivent bénéficier d'une sécurité maximale, c'est-à-dire dans la pratique s'effectuer via et sous le contrôle d'une carte à puce contenant toutes les données nécessaires à la sécurisation des échanges de données. Enfin, ces accès doivent pouvoir s'effectuer à partir d'un navigateur du commerce et être transparents pour un utilisateur, qui ne doit "voir" que la carte à puce comme unique interlocuteur, quel que soit le lieu de stockage de l'application.

Selon une première caractéristique importante du procédé, la carte à puce présente au système hôte, c'est-à-dire le terminal, un modèle de terminal virtuel, par exemple sous la forme d'une page en langage "HTML" (pour "HyperText Markup Language"), ou plus généralement en langage hypertexte, ou encore sous la forme d'un "applet", en langage "JAVA" (marque déposée), ce qui permet à l'utilisateur de choisir une application particulière parmi celles disponibles et proposées par la carte à puce. De ce fait, le terminal se trouve donc banalisé et supporte une pluralité d'applications. Le système hôte est vu

comme un périphérique de la carte à puce, et il met à sa disposition des ressources matérielles, tels un écran de visualisation, un clavier, etc.

Pour ce faire, on prévoit une couche de logiciel de communication spécifique dans la carte à puce et son pendant dans le terminal. Le terme "spécifique" doit être entendu comme spécifique au procédé de l'invention. En effet, ces couches de communications, dites spécifiques, sont banalisées quelle que soit l'application considérée. Elles n'interviennent que dans le processus d'échange de données bidirectionnel entre la carte à puce et le terminal, d'une part, et la carte à puce et le réseau.

Les couches logicielles de communication spécifiques comprennent notamment des composants logiciels, dits "agents intelligents", permettant en particulier des conversions de protocoles. Il existe des agents intelligents appareillés dans les couches de communication spécifiques respectives associées au terminal et à la carte à puce. Selon le procédé de l'invention, il s'établit des sessions entre agents intelligents appareillés.

Selon une deuxième caractéristique importante, le procédé de l'invention rend possible l'activation d'applications de type conventionnel, c'est-à-dire du type "CGA" précité, localisées dans une carte à puce, sans devoir les modifier en quoi que ce soit.

Pour ce faire, on prévoit un ou plusieurs agents intelligents dits traducteurs de script, qui reçoivent des requêtes d'un navigateur et les traduisent en ordres "APDU" compréhensibles par l'application de type "CGA". Cette caractéristique technique permet d'implanter dans une carte à puce, dont l'architecture est conforme au procédé de l'invention, un mécanisme similaire à la fonction dite "CGI" (pour "Common Gateway Interface") implantée dans les serveurs "WEB" classique.

Enfin, selon l'une des caractéristiques les plus importantes du procédé de l'invention, en mettant en œuvre les fonctions et mécanismes précités, celui-ci permet d'accéder à des ressources informatiques réparties sur un réseau de transmission de données auquel est connecté le terminal, notamment le réseau Internet ou un réseau de type équivalent (intranet,

extranet), sans que l'utilisateur ait à se soucier de leurs emplacements. Dans ce qui suit, comme il a été indiqué, ces ressources seront appelées "objets virtuels", statiques ou dynamiques.

Pour ce faire, il est mis en œuvre un agent intelligent traducteur de script dédié à cette tâche, coopérant avec les autres agents intelligents présents dans le terminal et/ou la carte à puce. Cet agent permet de définir les objets virtuels auxquels la carte à puce peut accéder, et de ce fait également l'utilisateur (ou porteur de la carte à puce), d'une part, et fournit au navigateur interrogateur, via la carte à puce, des méthodes permettant d'accéder à ces objets virtuels d'autre part.

L'invention a donc pour objet principal un procédé pour accéder à un objet localisé dans un système informatique connecté à un réseau de type Internet par l'intermédiaire d'un navigateur de type "WEB", ledit navigateur étant compris dans un terminal, muni d'un lecteur de carte à puce et connecté audit réseau de type Internet, caractérisé en ce qu'il comprend au moins les phases suivantes :

a/ une première phase préliminaire consistant à implanter, dans ladite carte à puce, une première pièce de logiciel, constituant une couche protocolaire de communication spécifique ;

b/ une deuxième phase préliminaire consistant à implanter, dans ledit terminal, une seconde pièce de logiciel, constituant une couche protocolaire de communication spécifique et formant interface avec au moins ledit navigateur "WEB" ;

– en ce que lesdites première et seconde pièces de logiciel spécifique comprennent en outre au moins une paire de premières entités logicielles appariées, chacune desdites entités coopérant l'une avec l'autre de manière à permettre l'établissement d'une session d'échanges de données bidirectionnels entre ledit terminal et ladite carte à puce, de manière à ce que ladite carte à puce offre la fonctionnalité d'un serveur "WEB" ;

– en ce qu'il est procédé à une troisième phase préliminaire consistant en l'implantation dans ladite carte à puce d'un système de gestion de fichiers dit

virtuel, organisé en répertoires, sous-répertoires et fichiers élémentaires, une partie au moins desdits fichiers élémentaires étant constitués par des fichiers élémentaires dits virtuels associés chacun à l'un desdits objets et contenant des données de description permettant au moins de localiser celui-ci et de générer des données de méthode d'accès ;

- en ce qu'il comprend une quatrième phase préliminaire consistant à implanter d'au moins une deuxième entité logicielle, apte à interpréter une suite d'instructions et à la traduire en une suite d'ordres, de manière à coopérer avec ledit système de gestion de fichiers virtuel et ladite seconde pièce de logiciel spécifique ;

– et en ce qu'il comprend au moins les étapes suivantes :

1/ établissement d'une session d'échanges de données entre ledit terminal et ladite carte à puce, par l'intermédiaire d'une desdites paires de premières entités logicielles appariées, de manière à transmettre une requête destinée à ladite deuxième entité logicielle coopérant avec ledit système de gestion de fichiers dit virtuel ;

2/ élaboration et transmission audit navigateur "WEB", par l'intermédiaire de ladite deuxième entité logicielle, et via lesdites première et seconde pièces de logiciel, formant lesdites couches protocolaires de communication spécifiques, d'une réponse à ladite requête comprenant une liste desdits objets accessibles via ladite carte à puce ;

3/ sélection par ledit navigateur "WEB" d'un desdits objets accessibles et envoi à ladite deuxième entité logicielle d'une requête supplémentaire pour obtenir une instance de celui-ci dans ledit terminal ;

4/ génération par cette ladite entité logicielle desdites données de méthode d'accès, en fonction desdites données de description ; et

5/ recherche dudit objet par utilisation desdites données de localisation et de méthode d'accès, et création d'une instance de cet objet dans ledit terminal.

L'invention a encore pour objet une architecture pour la mise en œuvre de ce procédé.

L'invention va maintenant être décrite de façon plus détaillée en se référant aux dessins annexés, parmi lesquels :

- les figures 1A et 1B illustrent schématiquement les architectures matérielles et logiques, respectivement, d'un exemple de système d'application à base de carte à puce selon l'art connu ;
- la figure 2 illustre schématiquement un exemple de système d'application à base de carte à puce selon l'invention, cette dernière agissant en tant que serveur "WEB" ;
- la figure 3 illustre de façon simplifiée l'architecture logique d'un système dans lequel la carte à puce comprend des agents intelligents ;
- la figure 4 illustre une architecture de système conforme à l'invention, dans laquelle la carte à puce comprend des agents intelligents traducteurs de scripts ;
- la figure 5 est un diagramme illustrant schématiquement les principales phase d'échanges entre un navigateur et une carte à puce présentant l'architecture de la figure 4 ;
- la figure 6 est un diagramme illustrant schématiquement un aspect essentiel du procédé selon l'invention par lequel il est possible d'accéder à des objets virtuels répartis sur un réseau de type Internet via une carte à puce et un navigateur de type "WEB" ;
- la figure 7 illustre schématiquement l'organisation d'un système de gestion de fichiers dit virtuel pour la mise en œuvre de cet aspect du procédé de l'invention ;
- la figure 8 est un exemple d'architecture comprenant un système de gestion de fichier virtuel selon la figure 7 ;
- les figures 9 à 15 sont des diagrammes illustrant schématiquement plusieurs mode de réalisation du procédé selon l'invention.

Avant de décrire le procédé d'activation d'applications localisées dans une carte à puce selon l'invention et de détailler une architecture pour sa mise en œuvre, il apparaît tout d'abord utile de rappeler brièvement les caractéristiques principales des protocoles de communication sur les réseaux.

L'architecture des réseaux de communication est décrite par diverses couches. A titre d'exemple, le standard "OSI" ("Open System Interconnection"), défini par l' "ISO", comporte sept couches qui vont des couches dites basses (par exemple la couche dite "physique" qui concerne le support de transmission physique) aux couches dites hautes (par exemple la couche dite d' "application"), en passant par des couches intermédiaires, notamment la couche dite de "transport". Une couche donnée offre ses services à la couche qui lui est immédiatement supérieure et requiert de la couche qui lui immédiatement inférieure d'autres services, via des interfaces appropriées. Les couches communiquent à l'aide de primitives. Elles peuvent également communiquer avec des couches de même niveau. Dans certaines architectures, l'une ou l'autre de ces couches peut être inexistante.

Dans un environnement de type Internet, les couches sont au nombre de cinq, et de façon plus précise, en allant de la couche supérieure à la couche inférieure : la couche d'application ("http", "ftp", "e-mail", etc.), la couche de transport ("TCP"), la couche d'adressage de réseau ("IP"), la couche de liens de données ("PPP", "Slip", etc.) et la couche physique.

Ces rappels étant effectués, on va maintenant décrire une architecture de système d'application à base de carte à puce autorisant celle-ci à agir comme un serveur "WEB". Un exemple d'une telle architecture est représenté schématiquement sur la figure 2. Les éléments communs aux figures 1A et 1B portent les mêmes références et ne seront re-décrits qu'en tant que de besoin. Pour simplifier le dessin, on n'a pas représenté les divers périphériques connectés au terminal (figure 1A : écran 5 et clavier 6, par exemple).

A l'exception de couches logicielles de protocole de communication spécifiques, référencées 13 et 23a, respectivement implantées dans le terminal 1 et la carte à puce 2*a*, les autres éléments, matériels ou logiciels, sont communs à l'art connu.

Le terminal 1 comprend des circuits 11 d'accès au réseau *RI*, constitués par exemple d'un modem pour le réseau Internet ou d'une carte réseau pour un

réseau local. Ces circuits regroupent les couches logicielles inférieures C_1 et C_2, correspondant aux couches "physique" et de "lien de données".

On a également représenté les couches supérieures C_3 et C_4, correspondant aux couches "d'adressage de réseau" ("IP", dans le cas d'Internet) et de "transport" ("TCP"). La couche supérieure d'application ("http", "ftp", "e-mail", etc.) n'a pas été représentée.

L'interface entre les couches inférieures, C_1 et C_2, et les couches supérieures, C_3 et C_4, est constituée par une couche logicielle généralement appelée "driver couches basses". Les couches supérieures, C_3 et C_4, s'appuient sur cette interface et sont mises en œuvre par l'intermédiaire de bibliothèques de fonctions spécifiques ou bibliothèques réseau 14, avec lesquelles elles correspondent. Dans le cas du réseau Internet, "TCP/IP" est mis en œuvre au moyen de bibliothèques dites de "sockets".

Cette organisation permet à un navigateur 10 (figure 1A) de poser des requêtes vers un serveur 4 (figure 1A), pour la consultation de pages "WEB" (protocole "HTTP"), pour le transfert de fichiers (protocole "FTP") ou l'envoi de courrier électronique (protocole "e-mail"), ce de façon tout à fait classique.

Le terminal 1 comprend également un lecteur de carte 3, intégré ou non. Pour communiquer avec la carte à puce 2*a*, le lecteur de carte englobe également deux couches basses, CC_1 (couche physique) et CC_2 (couche de lien de données), jouant un rôle similaire aux couches C_1 et C_2. Les interfaces logicielles avec les couches CC_1 et CC_2 sont décrites, par exemple, par la spécification "PC/SC" ("part 6, service provider"). Les couches elles-mêmes, CC_1 et CC_2, sont notamment décrites par les normes ISO 7816-1 à 7816-4, comme il a été rappelé.

Une couche logicielle supplémentaire 16 forme interface entre les couches applicatives (non représentées) et les couches inférieures, CC_1 et CC_2. La fonction principale dévolue à cette couche est une fonction de multiplexage/démultiplexage.

Les communications avec la carte à puce 2*a* s'effectuent selon un paradigme similaire à celui utilisé pour la manipulation de fichiers dans un système d'exploitation du type "UNIX" (marque déposée) : OUVRIR ("OPEN"), LIRE ("READ"), ECRIRE ("WRITE"), FERMER ("CLOSE"), etc.

Du côté de la carte à puce 2*a*, on retrouve une organisation similaire, à savoir la présence de deux couches basses, référencées CCa_1 (couche physique) et CCa_2 (couche de lien de données), ainsi qu'une couche d'interface 26a, tout à fait similaire à la couche 16.

Selon une première caractéristique importante, on prévoit, de part et d'autre, c'est-à-dire dans le terminal 1 et dans la carte à puce 2*a*, deux couches de protocoles spécifiques : 13 et 23*a*, respectivement.

Dans le terminal 1, la couche spécifique 13 s'interface aux "drivers couches basses" 15, aux bibliothèques 14 des couches réseau, C_3 et C_4, et aux couches protocolaires du lecteur de carte 3, c'est-à-dire les couches inférieures, CC_1 et CC_2, via la couche de multiplexage 16. La couche spécifique 13 permet le transfert des paquets réseaux de et vers la carte à puce 2*a*. En outre, elle adapte les applications existantes telles le navigateur Internet 10 (figure 2), le courrier électronique, etc., pour des utilisations mettant en œuvre la carte à puce 2*a*.

Du côté de la carte à puce 2*a*, on retrouve une organisation tout à fait similaire constituée par une instance supplémentaire de la couche spécifique, référencée 23a, pendant de la couche 13.

De façon plus précise, les couches spécifiques, 13 et 23a, sont subdivisées en trois éléments logiciels principaux :

- un module, 130 ou 230*a*, de transfert de blocs d'informations entre les couches 13 et 23a, via les couches conventionnelles CC_1, CC_2, CCa_1 et CCa_2 ;

- une ou plusieurs pièces de logiciel, dites "agents intelligents", 132 ou 232*a*, qui réalisent, par exemple, des fonctions de conversion de protocoles ;

- et un module de gestion de la configuration spécifique, 131 et 231*a*, respectivement ; module qui peut être assimilé à un agent intelligent particulier.

On retrouve donc, dans le terminal 1 et la carte à puce 2*a*, une pile protocolaire de communication entre les deux entités.

Les couches de niveau deux (couches de lien de données), CC_2 et CCa_2, assurent l'échange entre la carte à puce 2*a* et le terminal 1. Ces couches sont responsables de la détection et l'éventuelle correction d'erreurs de transmission. Différents protocoles sont utilisables, et à titre d'exemples non exhaustifs les suivants :

- la recommandation ETSI GSM 11.11 ;
- le protocole défini par la norme ISO 7816-3, en mode caractère T=0 ;
- le protocole défini par la norme ISO 7816-3, en mode bloc T=1 ;
- ou le protocole défini par la norme ISO 3309, en mode trame "HDLC" (pour "High-Level Data Link Control procedure" ou procédure de commande de liaison à haut niveau).

Dans le cadre de l'invention, on utilisera de préférence le protocole ISO 7816-3, en mode bloc.

De façon connue en soi, à chaque couche de protocole, il est associé un certain nombre de primitives qui permettent les échanges de données entre couches de même niveau et d'une couche à l'autre. A titre d'exemple, les primitives associées à la couche de niveau deux sont du type "demande de données" ("*Data.request*") et "envoi de données" par la carte ("*Data.response*"), ainsi que "confirmation de données" ("*Data.confirm*"), etc.

De façon plus spécifique, les couches 13 et 23a sont chargées du dialogue entre la carte à puce 2*a* et l'hôte, c'est-à-dire le terminal 1. Ces couches permettent l'échange d'informations entre un utilisateur (non représenté) du terminal 1 et la carte à puce 2*a*, par exemple via des menus déroulants sous la forme d'hypertexte au format "HTML". Elles permettent aussi la mise en place d'une configuration adaptée pour l'émission et/ou la réception de paquets de données.

Comme il a été indiqué ci-dessus, les couches comprennent trois entités distinctes.

La première couche, 130 ou 230*a*, est essentiellement constituée par un multiplexeur logiciel. Elle permet l'échange d'informations entre la carte à puce 2*a* et le terminal hôte 1, sous la forme d'unités de données de protocole. Elle joue un rôle similaire à celui d'un commutateur de paquets de données. Ces unités sont émises ou reçues via la couche de niveau 2 (couche de liens de données). Ce protocole particulier de communication permet de mettre en communication au moins une paire d' "agents intelligents". Le premier agent de chaque paire, 132, est situé dans la couche 13, côté terminal 1, le second, 232*a*, est situé dans la couche 23a, côté carte à puce 2*a*. Une liaison entre deux "agents intelligents" est associée à une session. Une session est un échange de données bidirectionnel entre ces deux agents.

Un agent intelligent peut réaliser tout ou partie de fonctions des couches de niveau trois et quatre, en fonction de la configuration mise en œuvre par le terminal 1.

Un agent intelligent particulier est identifié avantageusement par un nombre entier, par exemple sur 16 bits (nombre compris entre 0 et 65535). Cet identificateur est utilisé, par exemple, dans une unité de donnée de protocole constituant une référence de destination et une référence de source.

Il existe deux grandes catégories d'agents intelligents : les agents de type "serveur", qui sont identifiés par une référence fixe, et les agents de type "client", qui sont identifiés par une référence variable, délivrée par le module de gestion de configuration, 131 ou 231a.

Le processus d'ouverture d'une session est habituellement le suivant : un agent intelligent de type "client" ouvre la session vers un agent intelligent de type "serveur". Les couches 130 et 230*a* gèrent des tables (non représentées) qui contiennent la liste des agents intelligents présents, côté terminal hôte 1 et carte à puce 2*a*.

Les agents intelligents sont associés à des propriétés ou attributs particuliers. Pour fixer les idées, et à titre d'exemple non limitatif, les six propriétés suivantes sont associées aux agents intelligents :

- "hôte" : agent localisé dans le terminal ;

- "carte" : agent localisé dans la carte à puce ;
- "local" : agent ne communiquant pas avec le réseau ;
- "réseau" : agent communiquant avec le réseau ;
- "client" : agent qui initialise une session ;
- "serveur" : agent qui reçoit une demande de session.

Les agents intelligents permettent d'échanger des données (de l'hypertexte par exemple), mais également de déclencher des transactions réseau.

Les modules de gestion de configuration, 131 et 231*a*, respectivement, sont assimilables, comme il a été indiqué, à des agents intelligents particuliers. Par exemple, le module 131, côté terminal hôte 1, gère notamment des informations relatives à la configuration de ce terminal (modes de fonctionnement), liste des autres agents intelligents présents, etc. Le module 231*a*, côté carte à puce 2*a*, a des fonctions analogues. Ces deux agents intelligents peuvent être mis en communication l'un avec l'autre pour établir une session.

Selon une caractéristique importante, la carte à puce 2*a* propose au système hôte, c'est-à-dire au terminal 1, un modèle de terminal virtuel. Pour ce faire, la carte à puce 2*a* se comporte comme un serveur "WEB".

La carte à puce 2*a* est "adressée" par le navigateur 10. Elle lui transmet alors une page de type "WEB" en langage "HTML", un "applet" ou toute autre pièce de logiciel. A titre d'exemple, la page "WEB" peut se présenter sous la forme d'une page d'accueil donnant un choix d'applications possibles et/ou d'hyperliens vers des serveurs extérieurs.

De façon pratique, la carte à puce 2*a* est avantageusement "adressée" par utilisation d'une adresse "URL" (pour "Universal Resource Locator") définissant un rebouclage sur le terminal 1 lui-même, et non un pointage sur un serveur externe. A titre d'exemple, la structure de cette "URL" est habituellement la suivante :

http://127.0.0.1:8080 (1),

dans lequelle 127.0.0.1 est l'adresse "IP" de rebouclage et 8080 est le numéro de port.

La figure 3 illustre de façon simplifiée l'architecture logique d'un système dont la carte à puce 2*a* comprend des agents intelligents, dont deux seulement ont été représentés : un agent intelligent de type non précisément défini $232a_2$ et un agent intelligent $232a_1$, de type dit "WEB". La pile logique comprend, les couches de protocole inférieures, référencées 200*a*, répondant aux normes ISO 7816-3 (figure 2 : CCa_1 et CCa_2), le gestionnaire de commandes "APDU" $201a_1$, et le multiplexeur de paquets 230*a*, ce dernier étant interfacé aux agents intelligents, notamment l'agent intelligent "WEB" $232a_1$.

Du côté terminal, il existe deux piles, l'une communiquant avec le réseau Internet *RI*, l'autre avec la carte à puce 2*a*. La première pile comprend les organes 11 (figure 2 : C_1 et C_2) d'accès au réseau (normes OSI 1 et 2) et les couches de protocole "TCP/IP" (figure 2 : C_3 et C_4), référencées 100. Ces dernières couches sont interfacées avec le navigateur "WEB" 10. L'autre pile comprend les couches de protocole inférieures, référencées 101, répondant aux normes ISO 7816-3 (figure 2 : C_1 et C_2), le gestionnaire 102 d'ordres "APDU" et le multiplexeur de paquets 130, ce dernier étant interfacé avec des agents intelligents, dont un seul 132, est représenté. Ce dernier, que l'on supposera de "type réseau", peut en outre communiquer, d'une part avec le navigateur 10, via les couches "TCP/IP" 100, d'autre part avec le réseau Internet *RI*, via ces mêmes couches "TCP/IP" 100 et l'organe 11, d'accès au réseau *RI*.

Le gestionnaire d'ordres "APDU" 201*a* est également interfacé avec une ou plusieurs couches de niveau applications, que l'on appellera simplement applications. Ces applications sont, comme il a été indiqué, des applications de type conventionnel, que l'on a appelé "cardlet".

En résumé, la fonction "serveur WEB", fournie par la carte à puce 2*a*, peut être réalisée par l'association de l'agent intelligent "WEB" $232a_1$ dans la carte à puce et de l'agent réseau 132 dans le terminal 1.

La carte à puce 2*a* présente donc bien la fonctionnalité serveur "WEB". En outre, selon une caractéristique importante du procédé de l'invention, n'importe

quelle application conventionnelle, A_1 à A_n, du type "CGA" précité, peut être activée au travers de ce serveur "WEB", soit par le navigateur "WEB" 10 présent dans le terminal 1, soit par un navigateur éloigné, localisé en un point quelconque du réseau Internet *RI*. Selon le procédé de l'invention, les applications, A_1 à A_n, ne nécessitent pas d'être ré-écrites et sont mises en œuvre telles quelles.

Selon une autre caractéristique de l'invention, ces applications restent accessibles à un terminal de type conventionnel, c'est-à-dire conforme à l'art connu.

Pour répondre à ces exigences, la fonction serveur "WEB", offerte par la carte à puce 2*a*, inclut un mécanisme similaire à la fonction dite "CGI" (pour "Common Gateway Interface") implantée dans les serveurs "WEB" classiques.

Avant de décrire un exemple d'architecture conforme à l'invention, permettant de réaliser une fonction de ce type, au sein même de la carte à puce, il est utile de rappeler les principales caractéristiques d'un mode de fonctionnement "CGI".

Le "CGI" est une spécification de mise en œuvre, depuis un serveur "WEB", d'applications écrites pour les systèmes d'exploitation "UNIX" (marque déposée), "DOS", ou "WINDOWS" (marque déposée). A titre d'exemple, pour le système d'exploitation "UNIX", la spécification est "CGI 1.1" et pour le système d'exploitation "WINDOWS 95", la spécification est "CGI 1.3".

Toujours à titre d'exemple, une requête "HTTP" du type :

"http://www.host.com/cgi-bin/xxx.cgi" (2),

dans laquelle "host" se réfère à un système hôte (généralement éloigné), est interprétée par un serveur "WEB" comme l'exécution d'un script de commande, de type "CGI" nommé "xxx" et présent dans le répertoire "cgi-bin" de ce système hôte. Bien que le nom du répertoire puisse être *a priori* quelconque, par convention, c'est le nom donné au répertoire stockant les scripts de type "CGI". Un script est une suite d'instructions du système d'exploitation du système hôte dont le résultat final est transmis au navigateur "WEB" émetteur

de la requête précitée. Différents langages peuvent être utilisés pour écrire ce script, par exemple le langage "PERL" (marque déposée).

De façon pratique, la requête est habituellement affichée sur un écran informatique sous la forme d'un formulaire compris dans une page "HTML". Le langage "HTML" permet de traduire un formulaire en une adresse "URL". Le formulaire comporte un ou plusieurs champs, obligatoires ou non, qui sont remplis par un utilisateur à l'aide des moyens de saisie habituels : clavier pour le texte, souris pour les cases à cocher ou les boutons dits "radio", etc. Le contenu du formulaire (ainsi qu'éventuellement des informations et instructions dites "cachées") est émis à destination du serveur "WEB". Le code "HTML" de la page décrit la structure matérielle du formulaire (cadre, graphisme, couleur, et tout autre attribut), ainsi que la structure des champs de données à saisir (nom, longueur, type de données, etc.).

La transmission peut s'effectuer selon deux types de formats principaux. Un premier format utilise la méthode dite "POST" et un second la méthode dite "GET". Une information de type de format est présente dans le code de la page formulaire.

Ce mécanisme n'est cependant pas directement transposable à une carte à puce, même si celle-ci offre la fonctionnalité serveur "WEB" conformément à l'une des caractéristiques de l'invention.

On va maintenant décrire un exemple d'architecture permettant d'activer une application quelconque, de type conventionnel, via un serveur "WEB" sur la carte à puce 2*a*, par référence à la figure 4.

Lors d'une première étape, un utilisateur (non représenté) invoque depuis son navigateur "WEB" (figure 3 :10) une "URL" qui peut se présenter de la façon suivante :

"http://@carte:8080/xxx.html" (3),

dans laquelle "@carte" est une adresse IP de la carte à puce (par exemple l'adresse de rebouclage "127.0.01" précédemment décrite : voir formule (1)), et "xxx.html" est une page en langage "HTML" relative à une application particulière "xxx" offerte par la carte à puce.

Lors d'une deuxième étape, de la manière décrite précédemment, la carte à puce renvoie une page "HTML", par exemple de type formulaire.

Lors d'une troisième étape l'utilisateur renseigne les champs du formulaire et en transmet le contenu à la carte à puce, habituellement en cliquant sur un champ particulier, de type "bouton poussoir".

Les données sont alors émises et reçues par l'agent réseau 132. Les données traversent alors le multiplexeur de paquets 130 (qui constitue l'un des composants de la couche spécifique 13, côté terminal 1), le gestionnaire d'ordres "APDU" 102, les couches protocolaires 101, pour être transmises à la carte à puce 2*a*. Elles traversent ensuite les couches protocolaires 200*a*, le gestionnaire d'ordres "APDU" 201*a*, le multiplexeur de paquets 230*a* pour être reçues par l'agent "WEB" $232a_1$. Il s'établit donc une session logique entre les deux agents intelligents, comme explicité précédemment.

Il convient de remarquer que les données adressées à l'agent "WEB" $232a_1$ sont transportées, de façon conventionnelle en soi, sous formes d'ordres "APDU" destinés à l'application particulière "Multiplexeur de paquets". Le gestionnaire d'ordres "APDU" 201*a* sélectionne cette application de manière tout à fait similaire aux autres applications de type "CGA" présentes dans la carte à puce 2*a*, référencées A_1 à A_n. En d'autres termes, le multiplexeur de paquets 230a est vu par le gestionnaire d'ordres "APDU" 201*a* comme une application "CGA" ordinaire.

La requête "HTTP" est analysée par l'agent "WEB" $232a_1$ qui détecte une référence à un répertoire particulier, que l'on appellera ci-après par convention "cgi-smart", d'une part, et à une application particulière, par exemple "xxx" dans le cas de l'exemple décrit, d'autre part. Le chemin complet est donc, en l'occurrence "cgi-smart/xxx".

Selon une caractéristique importante du procédé de l'invention, l'entité ci-dessus désigne un script particulier associé à une application également particulière "xxx".

Lors d'une quatrième étape, le script est interprété par un agent intelligent dit "Agent traducteur de script", que l'on appellera "ATS" ci-après. Cette traduction peut être réalisée de différentes manières :

a/ par l'agent "WEB" 232*a*1 lui-même, qui est doté dans ce cas d'une double capacité ;

b/ par un agent traducteur de script unique capable de traduire l'ensemble des scripts présents dans la carte à puce 2*a* ;

c/ par un agent de script dédié que l'on appellera "ATSD" ci-après (un par script) ; ou

d/ par un agent "APDU" 2010*a* du gestionnaire d'ordres "APDU" 201*a*, qui est doté, dans ce cas, d'une double capacité.

L'agent "APDU" 2010*a* est une composante de la couche gestionnaire d'ordres "APDU" 201*a*. Cette dernière, comme il a été indiqué, est une couche capable de centraliser tous les ordres "APDU" émis et/ou reçus par le système, de sélectionner des applications, parmi A_1 à A_n, mais également d'offrir une interface de type agent intelligent. Elle est donc capable, selon l'une des caractéristiques du procédé, de communiquer avec tous les agents intelligents du système (via des sessions), que ces agents soient localisés dans le terminal 1 ou la carte à puce 2*a*.

Dans le cas *c/* ci-dessus, une session est ouverte entre l'agent "WEB" 232*a*1 et l'un des agents "ATSD".

La figure 4 illustre un exemple d'architecture pour laquelle les agents traducteurs sont du type "ATSD". Ils sont référencés *ATS*1 à *ATS*n et associés aux applications A_1 à A_n. L'application sélectionnée étant supposée être l'application A_i, la session s'établit entre l'agent "WEB" 232*a*1 et l'agent ATS_i.

Un agent traducteur de script génère une suite d'ordres "APDU". Une session est ouverte entre l'agent traducteur, par exemple l'agent ATS_i, et l'agent "APDU" 2010*a*. Les ordres sont alors émis vers l'agent "APDU" 2010*a*. Le gestionnaire d'ordres "APDU" 201*a* sélectionne l'application "CGA" A_i (par exemple l'application "PME") et lui transmet les ordres "APDU", ordres traduits

et donc conventionnels, qu'elle est en mesure de comprendre. Cette application est donc correctement activée, sans avoir à la modifier ou à la réécrire.

Les réponses de l'application "CGA" *Ai* sont transmises au gestionnaire d'ordres "APDU" 201*a*, à l'agent "APDU" 2010*a*, puis de nouveau à l'agent *ATSi* (et de façon plus générale à l'agent traducteur de script).

En fonction de la réussite ou de l'échec du déroulement du script, l'agent traducteur de script, par exemple l'agent *ATSi* dans l'exemple de la figure 4, élabore une page en langage "HTML" et la transmet via les différentes couches empruntées par la requête initiale, mais en sens inverse, ce pour être présentée sur l'écran de visualisation 5 (figure 1A).

Les différents cheminements sont représentés symboliquement sur la figure 4 par des traits pleins reliant les blocs fonctionnels ou en pointillés à l'intérieur de ces blocs.

La figure 5 résume de façon schématique les principales étapes du processus qui vient d'être décrit :

a/ transmission via le réseau Internet RI (ou à partir du terminal local : dans les deux cas à l'aide d'un navigateur conventionnel 10) d'une requête "HTTP", référencée RQ ;

b/ réponse du serveur "WEB" de la carte à puce 2a, sous la forme d'un formulaire, référencé FO ;

c/ transmission du formulaire rempli, sous la forme d'une nouvelle requête RQ ; et

d/ réponse sous la forme d'une page "HTML", référencée PR.

La réponse pourrait d'ailleurs consister en la transmission d'un fichier, ou d'une pièce de logiciel ou "Applet".

En mettant en œuvre les mécanismes et fonctions qui viennent d'être décrits, notamment la fonction serveur "WEB" et le recours à des agents intelligents traducteurs de scripts, selon une caractéristique essentielle, le procédé selon l'invention va permettre de définir un environnement virtuel, avantageusement

sécurisé par la carte à puce. Dans un mode de réalisation préféré, cet environnement est compatible avec les applications de type dit multimédia.

Cette dernière caractéristique est particulièrement avantageuse car les navigateurs "WEB" récents, mais de type tout à fait classique en soi, permettent, par nature, de construire des environnements multimédias (images animées, sons, etc.). Ils sont en effet associés à des outils logiciels, intégrés ou non, permettant de manipuler des fichiers multimédia (visionneuse, etc.). En tout état de cause, les navigateurs permettent de télécharger des fichiers de données multimédias, habituellement volumineux et de les stocker sur un disque dur, par exemple dans le terminal, ou sur un organe de stockage de masse similaire. On a notamment proposé des technologies permettant l'affichage en temps réel ou quasi-réel de séquences vidéos ou la reproduction de son, à partir de sites "WEB" du réseau Internet.

Cependant, une carte à puce, comme il a été rappelé, ne présente qu'une faible capacité de mémoire. En outre, elle n'autorise qu'un très faible débit de données lors des échanges. Il n'est donc pas possible d'enregistrer un grand nombre de fichiers de données dans celle-ci. Il n'est pas non plus envisageable, de façon pratique, de stocker des fichiers multimédias, à l'exception de très courtes séquences ou de séquences sonores codées sous un format particulier, tel que le codage "MIDI".

En dehors de ces limitations d'ordre technologique, il est aussi souhaitable de pouvoir avoir accès à des applications éloignées, tout en bénéficiant d'une sécurisation de haut niveau, que seule la mise en œuvre d'une carte à puce est apte à offrir.

Le procédé selon l'invention permet ce mode de fonctionnement. L'environnement virtuel multimédia sécurisé par carte à puce, selon un mode de réalisation préféré, permet de :

- définir des objets virtuels auxquels la carte à puce peut accéder ;
- fournir des méthodes d'accès à ces objets.

La figure 6 illustre schématiquement cet aspect essentiel du procédé selon l'invention.

Un utilisateur U_i interroge la carte à puce 2*a* grâce au navigateur "WEB" 10 compris dans le terminal 1. Selon un mécanisme qui va être précisé ci-après, grâce notamment à la fonction "serveur WEB" précédemment décrite, la carte à puce 2*a* va renvoyer au navigateur une liste d'objets dits virtuels Obv_i, *i* étant un indice arbitraire, auquel il a accès, c'est-à-dire de façon pratique pour lesquels la carte à puce 2*a* ou l'utilisateur U_i possède des droits d'accès. En effet, ces droits d'accès peuvent être liés strictement à la carte à puce 2*a* et immuables. Ils peuvent également être liés à un profil utilisateur, l'utilisateur U_i fournissant, par exemple, des données d'identification et un mot de passe. La carte à puce 2*a* effectue une vérification par comparaison avec des données d'une base de données de sécurité enregistrées dans une mémoire fixe et, si le résultat de la comparaison est positif, fournit une liste d'objets virtuels Obv_i associée au couple : "données d'identification - mot de passe". De façon connue en soi, cette première phase peut mettre en œuvre un procédé de chiffrage des données échangées entre le terminal et la carte à puce 2*a* ou mettre en œuvre un protocole de transmission sécurisé "HTTPS". La carte à puce 2*a* va également fournir une liste de méthodes d'accès aux objets virtuels Obv_i.

Les objets virtuels Obv_i, qui sont de type statique ou dynamique comme indiqué précédemment, peuvent être localisés indifféremment dans la carte à puce 2*a*, ou dans le terminal 1, ou, de façon plus générale, dans un système quelconque connecté au réseau Internet *RI*. Selon une caractéristique importante de l'invention, cet emplacement est "transparent" pour le navigateur 10, et donc pour l'utilisateur U_i, comme il va l'être montré.

Le procédé selon l'invention fait appel notamment à ce qui va être appelé ci-après un système de gestion de fichier virtuel, ou "SGFV", et un agent intelligent traducteur de script spécialisé, que l'on appellera "ATSDA/SGVF", dédié à cette tâche. Cet agent intelligent fournit la liste des objets virtuels Obv_i auxquels la carte à puce 2*a* peut accéder. Une adresse "URL" particulière est associée à chaque objet virtuel Obv_i. L'invocation de cette "URL" depuis le

navigateur "WEB" 10 permet d'instancier l'objet virtuel *Obvi*, au moyen d'une méthode d'appel déterminée, spécifique ou non à cet objet.

On va tout d'abord rappeler brièvement les principales caractéristiques d'un système de gestion de fichiers classique, appelé ci-après "SGF". Un tel système est utilisé pour stocker de l'information sur un support tel qu'un disque dur. L'information est mémorisée sous la forme d'un fichier. Un fichier, que ce soit des données pures ou des instructions de programme, est composé classiquement d'une suite de blocs de taille fixe. Un mécanisme bien connu permet d'obtenir la liste des blocs de mémoire qui constitue le fichier et leurs adresses dans la mémoire.

Un répertoire est un fichier particulier dont le contenu est une liste de descripteurs de fichier. Un tel descripteur comprend par exemple les éléments suivants :

- le nom du fichier ;
- la longueur du fichier ;
- la date de création ;
- une référence permettant de retrouver la liste des blocs du fichier (numéro du premier bloc, tableau des numéros de blocs, etc.) ; et
- et des attributs qui spécifient des propriétés particulières du fichier (répertoire, lecture, écriture, exécution, etc.).

Le premier répertoire est habituellement appelé répertoire racine. Un répertoire qui n'est pas racine est dit sous-répertoire. Le répertoire qui contient le descripteur d'un fichier donné est son répertoire père. L'adresse d'un fichier dans le "SGF" est donc une succession de noms de répertoires, depuis le répertoire racine jusqu'au répertoire père du fichier, ce qui définit un chemin. A titre d'exemple, un tel chemin se présente comme suit :

"/racine/répertoire1/répertoire2/nom_du_fichier" (4),

les chiffres 1 et 2 étant arbitraires, "racine" étant le nom du répertoire racine et "nom_de_fichier" un nom quelconque de fichier.

Pour une carte à puce, la norme ISO 7816-4 définit le répertoire racine dit "MF" (pour "Master File" ou fichier maître), des sous-répertoires dits "DF" (pour

"Dedicated Files" ou fichiers dédiés) et des fichiers élémentaires dits "EF" (pour "Elementary Files).

Dans le cadre de l'invention, le système de gestion de fichiers "SGFV", que l'on appellera virtuel, permet de définir des objets virtuels *Obvj* auxquels la carte à puce 2a peut avoir accès. Selon le procédé de l'invention, un objet virtuel *Obvj* est associé à un fichier élémentaire virtuel. Le contenu d'un fichier élémentaire virtuel est constitué par l'ensemble des informations qui permettent d'accéder à l'objet virtuel associé *Obvj* et d'en obtenir une instance dans le terminal 1.

De façon pratique, comme illustré schématiquement par la figure 7, le système "SGFV" peut constituer un sous-ensemble d'un système "SGF" classique, et, de façon plus précise, un "SGFV" est logé à l'intérieur d'un fichier élémentaire, tel que définit par la norme ISO 7816-4 précitée.

Un descripteur de fichier comportera généralement les éléments suivants :

- le nom du fichier ;
- la longueur du fichier ;
- la date de création ;
- une référence (avantageusement un nombre entier) qui permet de retrouver la liste des blocs du fichier (numéro du premier bloc, tableau de numéros de blocs, etc.) : un fichier virtuel est identifié par son nom ou cette référence unique ; et
- des attributs du fichier qui spécifient les références particulières du fichier : répertoire ou fichier élémentaire, virtuel ou non virtuel, direct ou indirect.

On appelle "objet virtuel direct", un objet qui est instancié depuis la carte à puce 2a. Il s'agit typiquement d'un objet virtuel *Obvj* statique qui peut être manipulé par le navigateur, par exemple affiché (image, etc.). On appelle " objet virtuel indirect", un objet virtuel *Obvj* qui est instancié depuis le navigateur 10, typiquement à l'aide d'un "applet".

La figure 8 illustre schématiquement l'architecture d'un système à carte à puce permettant d'instancier un objet virtuel *Obvj* localisé à un endroit quelconque du réseau Internet *RI*, via le navigateur 10 et la carte à puce 2a.

Les éléments communs aux figures précédentes portent les mêmes références et ne seront re-décrits qu'en tant que de besoin.

L'architecture illustrée sur la figure 8 est très similaire à celle de la figure 4. La différence essentielle est constituée par le fait que l'on a prévu un "SGFV" 8, stocké dans la carte à puce 2*a*, et un agent intelligent traducteur de script spécifique "ATSDA/SGFV", référencé 7. Le mode de fonctionnement est similaire à celui illustré par la figure 4 lorsque l'on désire accéder à une application particulière A_i. Il est donc inutile de le re-décrire en détail. Dans le cas présent l'application particulière est remplacée par le système de gestion de fichuer virtuel "SGFV" 8. On établit tout d'abord une session entre l'agent intelligent réseau 132 et l'agent intelligent "WEB" $232a_1$. Selon le mécanisme précédemment explicité, il s'établit ensuite une session entre l'agent "WEB" $232a_1$ et l'agent intelligent "ATSDA/SGFV" 7.

De façon pratique, l'agent intelligent "ATSDA/SGFV" 7 est accessible par des "URL", typiquement du type :

"http://www.host.com/cgi-smart/sgfv?" (5),

dans lesquelles "sgfv" est une application de type "CGI" associée à l'agent intelligent "ATSDA/SGFV" 7. La requête ci-dessus permet de parcourir l'arbre des répertoires et de "montrer" leur contenu au navigateur 10, au moyen d'une page "HTML". Les "feuilles "de l'arbre sont des fichiers élémentaires, virtuels ou non virtuels, associés à un hyperlien. La transmission dans le sens "carte à puce 2*a* - terminal 1" est réalisée de la manière qui a été explicitée en regard de la figure 4.

En d'autres termes, l'agent intelligent "ATSDA/SGFV" 7 associe à tout élément du "SGFV" 8, répertoire ou fichier élémentaire, une adresse "URL". L'adresse "URL" d'un répertoire désigne une page "HTML" qui contient la liste de ses éléments. L'adresse "URL" d'un fichier élémentaire permet de créer une instance de l'objet virtuel Obv_i associé à ce fichier virtuel.

Pour fixer les idées, si on utilise l'adresse "URL" (5) ci-dessus, on obtient une page "HTML" qui présente le contenu du répertoire racine au navigateur 10. Ce répertoire racine est constitué par un ensemble de sous-répertoires et de

fichiers comme illustré schématiquement par la figure 9. Sur cette figure, on a représenté un répertoire racine *rep*#0, au niveau supérieur, un fichier élémentaire réel *fe*#7 et un sous-répertoire réel *srep*#1, au niveau immédiatement inférieur, et un sous-répertoire virtuel *rep*#2 et un fichier élémentaire virtuel *fe*#5, au niveau le plus bas, tous deux dépendants du sous-répertoire réel *srep*#1, les numéros de référence étant purement arbitraires .

Lors d'une première phase, l'agent intelligent "ATSDA/SGFV" 7 transmet au navigateur 10, en réponse à la requête reçue, une page "HTML" (non représentée) montrant, sous une forme ou une autre, la structure hiérarchique du "SGFV" 8. La page est habituellement affichée sur un écran de visualisation (figure 1A : 5), par exemple sous la forme d'un menu. Chaque ligne du menu est constituée d'un hyperlien décrivant un sous-répertoire ou un fichier élémentaire. L'affichage peut être avantageusement sous forme graphique, associée ou non à un texte descriptif, le dessin de l'arbre de la figure 9 étant affiché sur l'écran précité. On peut encore afficher des icônes ou des formes complexes (par exemple en 3 dimensions), chacune étant associée à l'un des objets virtuels à instancier, et pouvant être représentative de leur nature (à titre d'exemple, une caméra représentant un fichier vidéo), associée ou non à un texte descriptif.

L'utilisateur U_j est invité à cliquer sur un hyperlien (sur un nœud ou sur une branche dans le cas d'un graphique). Par cette action, il va pouvoir obtenir une instance de l'objet virtuel Obv_j désiré.

Le système "SGFV" 8 est avantageusement enregistré dans une mémoire de type re-programmable comprise dans la carte à puce 2*a*, par exemple du type "EEPROM" (mémoire effaçable électriquement), comme illustré schématiquement sur la figure 10. Le "SGFV" 8 reproduit la structure de l'arbre de la figure 9.

Toujours dans l'exemple décrit, une fois obtenu la page de menu, obtenue lors d'une phase initiale, en cliquant sur une "URL" qui pourrait être typiquement la suivante :

"http://www.host.com/cgi-smart/file#5 (6),

l'utilisateur U_i obtient une instance de l'objet virtuel Obv_5 associé au fichier élémentaire référencé *fe*#5 sur la figure 10. De même, il aurait pu obtenir le contenu d'un sous-répertoire, le paramètre "file#5" étant remplacé par "*file#x*" dans (6), #*x* étant le numéro associé au sous-répertoire.

Les fichiers non virtuels sont enregistrés dans la carte à puce 2*a* et sont conformes au paradigme usuel qui gouverne les "SGF". Ils contiennent des données, telles que par exemple des clés, données utiles à l'agent intelligent "ATSDA/SGFV" 7.

Différentes conventions sont possibles quant à la définition des informations nécessaires à l'instanciation d'un objet virtuel Obv_i, et par exemple :

- un fichier virtuel de longueur nulle hérite des méthodes d'accès de son répertoire père ;
- un répertoire virtuel est associé à un fichier élémentaire virtuel dont le nom est imposé (par exemple "virtual"), et qui contient les méthodes d'accès de ce répertoire.

En effet, outre la liste des objets virtuels accessible Obv_i, un agent intelligent "ATSDA/SGFV" 7 doit également fournir une méthode d'accès à un objet virtuel donné Obv_i, ce à partir de tout ou partie des informations contenues dans un fichier virtuel élémentaire. La figure 11 illustre schématiquement ce processus.

Selon le procédé de l'invention, on prévoit deux méthodes d'accès, que l'on appellera directe et indirecte, respectivement, en fonction des attributs du fichier élémentaire virtuel considéré.

La méthode directe consiste en une description d'une chaîne d'agents intelligents mis en œuvre dans le processus permettant d'accéder à un objet virtuel Obv_i et d'en obtenir une instance dans le terminal. Lors de l'ouverture d'une session, un agent intelligent donné reçoit, de l'agent qui initie cette session, une liste de structures d'appel qui sera dénommée ci-après "méthode d'appel" ou encore "Method PDU" (pour "Method Protocol Data Unit").

Une structure d'appel comprend :

- un identificateur de l'agent intelligent avec lequel la session est ouverte ;
- des données, ou argument, nécessaires à son utilisation.

Le premier agent intelligent adressé par la liste précitée "consomme" une première structure d'appel qui lui est destinée. Il transmet le reste de la liste de structure à un agent intelligent suivant, avec lequel il établit une session, jusqu'à épuisement de la liste.

Pour fixer les idées, un exemple des différentes étapes d'échanges entre l'agent intelligent "ATSDA/SGFV" 7 et deux agents intelligents en cascade, $232a_m$ et $232a_n$, est illustré schématiquement par la figure 12. La liste de structure d'appel émise par l'agent intelligent "ATSDA/SGFV" 7 comporte en réalité deux sous-listes distinctes, repérées #1 et #2 dans leurs en-têtes, respectivement. La première est consommée par le premier agent intelligent, $232a_m$, et la seconde par le second agent intelligent, $232a_n$. Un agent intelligent, par exemple l'agent intelligent $232a_m$, est identifié par une référence, ou identificateur d'agent ("*Identificateur Agent* #1" ou "*Identificateur Agent* #2"). L'agent intelligent adressé, et avec lequel s'établit une session, retient la sous-liste qui lui est destinée grâce à l'en-tête ("*Structure d'Appel* #1 *ou "Structure d'Appel* #2"). Les arguments de la sous-liste qu'il retient ("*Argument#1*" ou "*Argument#1*") sont constitués par un ensemble de données utiles au bon fonctionnement de cet agent. A titre d'exemple, une donnée peut être un nom de fichier (non virtuel ou virtuel direct).

Un agent intelligent donné, par exemple l'agent intelligent $232a_m$, peut modifier le reste de la liste de structure d'appel avant de la transmettre à l'agent intelligent suivant, $232a_n$. Pour ce faire, il adresse cet agent intelligent, $232a_n$, et établit une session avec lui.

La méthode d'appel peut être avantageusement décrite au moyen du langage ASN.1 (Abstract Syntax Notation 1" de l'ISO)

La méthode d'accès directe permet en définitive d'instancier un objet virtuel Obv_i directement depuis la carte à puce 2*a*. Il s'agit *a priori* d'un objet

statique. L'objet instancié se présente habituellement sous la forme d'une page "HTML" ou d'un "applet" transmis au navigateur 10.

La seconde méthode d'accès, ou méthode d'accès indirect, est en réalité également une méthode d'accès direct, mais mise en œuvre à partir du terminal 1, et non plus de la carte à puce 2*a*. Cette méthode est essentiellement utilisée pour instancier des objets virtuels *Obv*$_i$ du type dynamique.

Selon cette variante du procédé, en réponse à un "URL" qui désigne un fichier élémentaire virtuel *fe#x*, l'agent intelligent "ATSDA/SGFV" 7 transmet au navigateur 10 une page "HTML" qui comporte un hyperlien pointant sur la méthode d'accès directe associée à l'objet virtuel *Obv*$_i$

Deux variantes peuvent être mises en œuvre.

La première variante consiste à utiliser un "applet". Le lien sur la méthode d'accès est alors un "applet" localisé à l'adresse "*@carte*", qui peut elle-même être désignée par :

- le nom (c'est-à-dire une "URL") d'un fichier non virtuel, enregistré sur la carte à puce 2*a* ;
- une "URL" qui désigne un fichier virtuel direct.

Un paramètre d'appel de cet "applet" est une liste de structure d'appel, par exemple codée en ASN.1 comme indiqué ci-dessus. L' "applet" contenu dans une page "HTLM" est téléchargé, depuis la carte à puce 2*a* ou le réseau Internet *RI*, vers le navigateur 10, puis exécuté par celui-ci de façon obligatoire (forçage). Cet "applet" établit une session avec un premier agent intelligent, arbitrairement référencé 232a_p. La connexion à cet agent intelligent 232a_p utilise, par exemple, un modèle d'échange de données du type client-serveur "TCP/IP" (c'est-à-dire la classe dite "socket JAVA"). L' "applet" se comporte comme un client "TCP/IP" et se connecte à un serveur "TCP/IP" (ce dernier étant également un agent intelligent) identifié par l'adresse de la carte et un port : "@carte : port".

La figure 13 illustre schématiquement les différentes phases des échanges permettant d'instancier un objet virtuel par la méthode indirecte. On a repris sur

cette figure 13 les paramètres de l'exemple précédemment décrit, en l'occurrence, le fichier élémentaire virtuel *fe*#5, ce qui se traduit par l'adresse "URL" de la configuration (6) ci-desus. On a supposé que l'adresse de la carte à puce à utiliser est "*@carte*" et le port 8080. La requête est transmise à l'agent intelligent "ATSDA/SGFV" 7 selon le processus précédemment décrit. Celui-ci renvoie au navigateur 10 une page "HTML" *P* constituée par un "applet". Dans un but de simplification du dessin, les différentes instructions de cet "applet" ont été résumées sur la figure 13 par la mention "*CODE applet*" placée entre les marqueurs <applet ...> et </applet>. De façon connue en soi, l' "applet" est associé à une classe "JAVA", que l'on a appelé arbitrairement "*tv.class*", pour "terminal virtuel". Le code comprend également des instructions indiquant l'adresse du premier agent intelligent de la structure de liste , référencé $232a_p$, et l'adresse et le port à utiliser, en l'occurrence l'adresse "*@carte*" et le port 8081. Il est à noter que cet agent intelligent $232a_p$ peut être localisé dans la carte à puce 2*a* ou dans le terminal 1.

Chaque agent intelligent, par exemple $232a_p$, exécute une tâche précise : déchiffrement d'un message chiffré, vérification de mots de passe et/ou de données de sécurité, conversion d'un fichier, d'un format à un autre, etc. Bien que l'on ait représenté qu'un seul agent intelligent, $232a_p$, comme dans le cas précédent (figure 12) on peut prévoir, en tant que de besoin, plusieurs agents intelligents en cascade. Comme précédemment également chaque agent intelligent, $232a_p$, consomme une partie de la structure de liste, celle qui lui est destinée, et transmet le reste, inchangé ou modifié, à l'agent intelligent suivant (non représenté).

Pour mieux illustrer la première variante, et pour fixer les idées, on va considérer que l'utilisateur U_i désire télécharger et exécuter un fichier audio, codé par exemple au format "MP3". Ce fichier constitue l'un des objets virtuels, ici référencé *FS*, proposés par la page de menu "HTML" transmise par l'agent intelligent "ATSDA/SGFV" 7, lors de la phase initiale. La figure 14 illustre schématiquement la séquence d'étapes permettant d'instancier un tel objet virtuel, référencé *FS*. On suppose que le navigateur 10 ne dispose pas d'un

lecteur approprié pour un tel format. Ce lecteur, référencé *LS*, va être cherché sur un site Internet, qui peut être distinct ou non du site où se trouve le fichier son *FS*.

Dans l'exemple décrit, la séquence d'étapes est la suivante.

a/ l'utilisateur U_i clique sur un hyperlien (texte, icône ou toute autre représentation graphique de l'objet à rechercher, c'est-à-dire le fichier *FS*) : une requête I_1 est transmise à la carte à puce 2*a* ;

b/ en réponse, R_1, une page "HTML" est transmise, par la carte à puce 2*a*, au terminal 1 et au navigateur 10 ;

c/ la page "HTML" reçue force le navigateur 10 à demander un "applet" : interrogation I_2 (dans le cas présent, il s'agit d'aller chercher le lecteur de son approprié *LS*) ;

d/ en réponse, R_2, le lecteur recherché *LS* est téléchargé et installé dans le terminal 1 ;

e/ le navigateur 10 adresse de nouveau la carte à puce 2*a*, requête I_3, en vue d'obtenir une instance du fichier audio *FS*; et

f/ en réponse, le navigateur 10 reçoit ce fichier audio *FS*, ce dernier pouvant être lu, c'est-à-dire joué par le terminal 1, qui dispose désormais du lecteur de son *LS* approprié.

Il est à noter que toutes les opérations sont transparentes pour l'utilisateur U_i, plus précisément pour le navigateur 10, qui ne "connaît" que la carte à puce 2*a*. Le lecteur *LS* (ou de façon plus générale un autre "applet") et/ou l'objet virtuel recherché, c'est-à-dire le fichier *FS* dans l'exemple, si leurs tailles avaient été compatibles avec la capacité de mémorisation de la carte à puce 2*a*, auraient d'ailleurs pu être enregistrés dans celle-ci (re-bouclages symbolisés par des traits en pointillés sur la figure 14). Le navigateur 10 ne connaît pas la localisation exacte des objets virtuels Obv_i. Seule la carte à puce 2*a*, de façon plus précise l'agent intelligent "ATSDA/SGFV" 7, connaît la localisation des objets virtuels de la liste du "SGFV" 8 et la méthode pour y accéder.

Dans une variante préférée du procédé, l'agent intelligent "ATSDA/SGFV" 7 connaît également la liste des seuls objets virtuels accessibles à un utilisateur U_i donné (autorisations). Il s'agit donc bien d'un système sécurisé. Le terme "sécurisé" doit être considéré dans son sens le plus large. A titre d'exemple, il concerne aussi bien des cartes à péage donnant accès à certaines ressources, en fonction d'un abonnement déterminé par exemple, ou des cartes assurant un accès sécurisé proprement dit à des ressources confidentielles, en fonction d'un niveau d'habilitation par exemple. Comme il a été indiqué, les ressources ou objets virtuels Obv_i peuvent être constituées par des transactions.

Ces remarques s'appliquent d'ailleurs aussi pour les méthodes d'accès direct. Cela constitue une caractéristique très importante du procédé selon l'invention.

Selon une seconde variante, illustrée schématiquement par la figure 15, on peut utiliser un hyperlien qui définit l'adresse "TCP/IP" du premier agent intelligent associé à la méthode d'accès. l'adresse est du type : "@Agent:AgentPort", avec "@Agent", l'adresse proprement dite de l'agent intelligent concerné et "AgentPort" le port de celui-ci. La liste "MethodPDU" est dans ce cas un paramètre d'une "URL". L'hyperlien sera associé, par exemple à une image ou un formulaire d'une page "HTML" P'.

Ainsi, à titre d'exemple, une "URL" ayant la structure suivante :

http://@Agent:AgentPort/MethodPDU?Value=*xx*... (7),

permet d'atteindre un agent intelligent qui se comporte comme un serveur "WEB TCP/IP", référencé arbitrairement $232a_q$. Cet agent intelligent $232a_q$ est localisé à l'adresse "@Agent:AgentPort" et reçoit la liste de structure d'appel "MethodPDU", avec le paramètre "Value=*xx*...".

Pour fixer les idées, on suppose que l'objet virtuel Obv_i est une image à afficher sur un écran (figure 1A :5), d'un format particulier, à l'aide du navigateur 10 et que celui-ci ne dispose pas d'un programme approprié pour cet affichage, généralement appelé visionneuse ou "viewer" selon la terminologie anglo-saxonne. Il s'agit, par exemple, d'un programme exécutable sous le système d'exploitation utilisé sur le terminal 1, de type "XXX.exe", avec "XXX" le nom de ce programme. L'action de cliquer sur l'hyperlien (7) ci-dessus

va permettre de rechercher ce programme exécutable, celui-ci pouvant être localisé dans le terminal 1 ou dans un système éloigné.

La différence entre les deux variantes de réalisation est que dans le premier cas, le navigateur se voit "forcé" de demander le chargement d'un "applet". Toutes les étapes sont réalisées automatiquement. Dans le second cas, l'utilisateur U_i est invité à cliquer sur un hyperlien ou à exécuter une action similaire.

A la lecture de ce qui précède, on constate aisément que l'invention atteint bien les buts qu'elle s'est fixés.

Il doit être clair cependant que l'invention n'est pas limitée aux seuls exemples de réalisations explicitement décrits, notamment en relation avec les figures 2 à 15.

En particulier, comme en ce qui concerne les autre agents intelligents traducteurs de script; la fonction de l'agent intelligent associé au système de gestion de fichier virtuel peut être remplie par un agent non dédié : l'agent "WEB" ou l'agent "APDU".

REVENDICATIONS

1. Procédé pour accéder à un objet localisé dans un système informatique connecté à un réseau de type Internet par l'intermédiaire d'un navigateur de type "WEB", ledit navigateur étant compris dans un terminal, muni d'un lecteur de carte à puce et connecté audit réseau de type Internet, caractérisé en ce qu'il comprend au moins les phases suivantes :

- a/ une première phase préliminaire consistant à implanter, dans ladite carte à puce (2*a*), une première pièce de logiciel (23*a*), formant une couche protocolaire de communication spécifique ;
- b/ une deuxième phase préliminaire consistant à implanter, dans ledit terminal (1), une seconde pièce de logiciel (13), formant une couche protocolaire de communication spécifique et formant interface avec au moins ledit navigateur "WEB" (10) ;

– en ce que lesdites première et seconde pièces de logiciel (13, 23*a*) comprennent en outre au moins une paire de premières entités logicielles appariées (132, 232*a*), chacune desdites entités (132, 232*a*) coopérant l'une avec l'autre de manière à permettre l'établissement d'une session d'échanges de données bidirectionnels entre ledit terminal (1) et ladite carte à puce (2*a*), de manière à ce que ladite carte à puce (2*a*) offre la fonctionnalité d'un serveur "WEB" ;

– en ce qu'il est procédé à une troisième phase préliminaire consistant en l'implantation dans ladite carte à puce (2*a*) d'un système de gestion de fichiers dit virtuel (8), organisé en répertoires (*rep*#0), sous-répertoires (*srep*#1, *srep*#2) et fichiers élémentaires (*fe*#7, *fe*#5), une partie au moins desdits fichiers élémentaires étant constitués par des fichiers élémentaires dits virtuels (*fe*#5), associés chacun à l'un desdits objets (*Obvi*) et contenant des données de description permettant au moins de localiser celui-ci et de générer des données constituant une méthode d'accès ;

- en ce qu'il comprend une quatrième phase préliminaire consistant à implanter dans ladite carte à puce (2*a*) au moins une deuxième entité logicielle

(7), apte à interpréter une suite d'instructions et à la traduire en une suite d'ordres, de manière à coopérer avec ledit système de gestion de fichiers virtuel (8) et ladite seconde pièce de logiciel spécifique (23*a*) ;

– et en ce qu'il comprend au moins les étapes suivantes :

- 1/ établissement d'une session d'échanges de données entre ledit terminal (1) et ladite carte à puce (2*a*), par l'intermédiaire d'une desdites paires de premières entités logicielles appariées (132, 232a_1), de manière à transmettre une requête destinée à ladite deuxième entité logicielle (7) coopérant avec ledit système de gestion de fichiers dit virtuel (8) ;
- 2/ élaboration et transmission audit navigateur "WEB" (10), par l'intermédiaire de ladite deuxième entité logicielle (7), et via lesdites première (13) et seconde (23*a*) pièces de logiciel, d'une réponse à ladite requête comprenant une liste desdits objets (*Obvi*) accessibles via ladite carte à puce (2*a*) ;
- 3/ sélection par ledit navigateur "WEB" (10) d'un desdits objets accessibles et envoi à ladite deuxième entité logicielle (7) d'une requête pour obtenir une instance de celui-ci dans ledit terminal ;
- 4/ génération par ladite entité logicielle (7) desdites données constituant une méthode d'accès, en fonction desdites données de description ; et
- 5/ recherche dudit objet (*Obvi*) par utilisation desdites données de localisation et de méthode d'accès, et création d'une instance de cet objet (*Obvi*) dans ledit terminal (1).

2. Procédé selon la revendication 1, caractérisé en ce que ledit lecteur de carte à puce (3) et ladite carte à puce (2a) comprennent des première et deuxième piles protocolaires pour lesdites transmissions de données, définies par la norme ISO 7816, chacune comprenant au moins des couches protocolaires de communication logicielles (101, 200a), dites basses, de manière à permettre lesdits échanges de données entre ladite carte à puce (2a) et ledit terminal (1), ces couches formant interface avec lesdites première (13) et seconde (23a) pièces de logiciel spécifique formant lesdites couches protocolaires de communication spécifiques, respectivement, et en ce que ces

pièces de logiciel (13, 23a) comprennent chacune deux entités supplémentaires constituées d'un module de transfert de données (130, 230a), formant interface avec lesdites couches basses (101, 200a) des première et deuxième piles protocolaires, et d'un module de gestion (131, 231a), et en ce que lesdites premières entités de chaque paire sont constituées de modules logiciels, dits agents intelligents (132, 232a1) établissant lesdites sessions.

3. Procédé selon la revendication 2, caractérisé en ce que, ladite suite d'instructions à interpréter étant constituée par un script, ladite deuxième entité logicielle (7) coopérant avec ledit système de gestion de fichiers virtuel (8) est constituée par un module logiciel dit agent intelligent traducteur de script.

4. Procédé selon la revendication 3, caractérisé en ce que ledit agent traducteur de script (7) coopérant avec ledit système de gestion de fichiers virtuel (8) est confondu avec ledit agent intelligent ($232a_1$) établissant lesdites sessions d'échanges de données entre ledit terminal (1) et ladite carte à puce (2*a*).

5. Procédé selon la revendication 3, caractérisé en ce que lesdites applications sont d'un type devant être activé à l'aide d'un jeu d'ordres dits "ADPU", définis par la norme ISO 7816, ledit terminal (1) et ladite carte à puce (2*a*) comprennent une entité logicielle supplémentaire (102, 201*a*), dite gestionnaire d'ordres "ADPU", en ce que ladite interface entre lesdites couches basses (101, 200*a*) des première (101) et deuxième (200*a*) piles protocolaires, d'une part, et lesdites première (13) et seconde (23*a*) pièces de logiciel spécifique (23*a*), d'autre part, s'effectue par l'intermédiaire desdits gestionnaires d'ordres (102, 200*a*).

6. Procédé selon la revendication 5, caractérisé en ce que ledit gestionnaire d'ordres (201*a*) de la carte à puce (2*a*) comprend en outre une entité logicielle du type agent intelligent (2010*a*) permettant l'établissement d'une session d'échanges de données bidirectionnels avec d'autres agents intelligents, cet agent (2010*a*) étant dit agent intelligent gestionnaire d'ordres, et en ce que

ledit agent traducteur de script (7) coopérant avec ledit système de gestion de fichiers virtuel (8) est confondu avec ledit agent intelligent gestionnaire d'ordres (2010*a*).

7. Procédé selon la revendication 1, caractérisé en ce que ladite deuxième entité logicielle (7) coopérant avec ledit système de gestion de fichier virtuel (8) associe, à chacun desdits répertoires (*rep*#0), sous-répertoires (*srep*#1, *srep*#2) et fichiers élémentaires (*fe*#7, *fe*#5), une adresse de type Internet dite "URL", en ce que ladite étape 2/ de réponse consiste en l'envoi d'une page en langage "HTML" comprenant des hyperliens pointant sur lesdites adresses "URL", et en ce que ladite étape 3/ de sélection consiste à actionner l'un de ces hyperliens de manière à accéder à l'un desdits répertoires (*rep*#0), sous-répertoires (*srep*#1, *srep*#2) ou fichiers élémentaires (*fe*#7, *fe*#5) ; l'accès à un répertoire (*rep*#0) ou à un sous-répertoire (*srep*#1, *srep*#2) occasionnant la transmission vers ledit navigateur "WEB" (10) d'une page en langage "HTML" présentant son contenu, et l'accès à un desdits fichiers élémentaires virtuels (*fe*#5) occasionnant ladite recherche de l'objet (*Obvi*) associé à ce fichier élémentaire virtuel (*fe*#5) et la création d'une instance de cet objet (*Obvi*) dans ledit terminal (1).

8. Procédé selon la revendication 7, caractérisé en ce qu'il est prévu au moins un agent intelligent traducteur de script supplémentaire ($232a_m$, $232a_m$) destiné à l'interprétation d'un script prédéterminé, de manière à remplir une tâche prédéterminée, et en ce qu'il comporte une première méthode d'accès aux dits objets (*Obvi*), dite directe, consistant à décrire lesdits agents intelligents supplémentaires mis en œuvre pour accéder aux dits objets (*Obvi*) et comprenant au moins les étapes suivantes :

- a/ création d'une structure d'appel comprenant au moins un élément, chaque élément comportant un identificateur d'un desdits agents intelligents supplémentaires ($232a_m$, $232a_m$) et des données, dites argument, utilisées par celui-ci pour l'exécution de ladite tâche ;

- b/ l'établissement d'une session entre un premier agent intelligent supplémentaire (232a_m) et ledit agent intelligent (7) coopérant avec ledit système de gestion de fichier virtuel (8), cet agent intelligent supplémentaire (232a_n) étant sélectionné par l'identifiant d'un premier élément de ladite structure d'appel et retenant un élément de ladite structure qui lui est destiné ;
- c/ lorsque ladite structure comporte plusieurs éléments, établissement d'une session entre ledit premier agent intelligent supplémentaire (232a_m), et un autre agent intelligent supplémentaire (232a_n), et transmission des éléments restants de la structure d'appel, cet autre agent intelligent supplémentaire (232a_n) étant sélectionné par ledit identifiant et retenant un élément de ladite structure qui lui est destiné ; et
- d/ répétition de l'étape c/ jusqu'à épuisement des éléments constituant ladite structure d'appel.

9. Procédé selon la revendication 7, caractérisé en ce qu'il comporte une seconde méthode d'accès, dite indirecte, comprenant au moins une étape consistant, en réponse à une requête transmise par ledit navigateur "WEB" (10) à ladite deuxième entité logicielle (7) coopérant avec ledit système de gestion de fichier virtuel (8), désignant un desdits fichiers élémentaires virtuels (fe#5), à transmettre au navigateur "WEB" (10) une page en langage "HTML" (*P*) comportant un hyperlien pointant sur un enregistrement comportant des données nécessaires à la génération de ladite méthode d'accès.

10. Procédé selon la revendication 9, caractérisé en ce que ladite page en langage "HTML" (*P*) comprend une pièce de logiciel en langage "JAVA", chargée depuis ladite carte à puce (2*a*) ou depuis ledit réseau de type Internet (*RI*), sous la commande dudit agent intelligent (7) coopérant avec ledit système de gestion de fichier virtuel (8), et exécutée par ledit navigateur "WEB" (10), de manière à retrouver des données nécessaire à la génération de ladite méthode d'accès dans un fichier élémentaire non virtuel dudit système de gestion de fichiers virtuel (8) ou par l'intermédiaire d'une adresse contenue dans un desdits fichiers élémentaires virtuels (*fe*#5).

11. Procédé selon la revendication 9, caractérisé en ce qu'il est prévu au moins un agent intelligent traducteur de script supplémentaire (232a_p) destiné à l'interprétation d'un script prédéterminé, de manière à remplir une tâche prédéterminée, en ce que ladite seconde méthode d'accès indirect comprend au moins les étapes subséquentes suivantes consistant à décrire lesdits agents intelligents supplémentaires mis en œuvre pour accéder aux dits objets (*Obvi*) :

- a/ création d'une structure d'appel comprenant au moins un élément, chaque élément comportant un identificateur d'un desdits agents intelligents supplémentaires (232a_p) et des données, dites argument, utilisées par celui-ci pour l'exécution de ladite tâche ;
- b/ l'établissement d'une session entre un premier agent intelligent supplémentaire (232a_p) et ledit agent intelligent (7) coopérant avec ledit système de gestion de fichier virtuel (8), cet agent intelligent supplémentaire (232a_p) étant sélectionné par l'identifiant d'un premier élément de ladite structure d'appel et retenant un élément de ladite structure qui lui est destiné ;
- c/ lorsque ladite structure comporte plusieurs éléments, établissement d'une session entre ledit premier agent intelligent supplémentaire (232a_m), et un autre agent intelligent supplémentaire (232a_n), et transmission des éléments restants de la structure d'appel, cet autre agent intelligent supplémentaire (232a_n) étant sélectionné par ledit identifiant et retenant un élément de ladite structure qui lui est destiné ; et
- d/ répétition de l'étape c/ jusqu'à épuisement des éléments constituant ladite structure d'appel.

12. Procédé selon la revendication 10, caractérisé en ce qu'il est prévu au moins un agent intelligent traducteur de script supplémentaire (232a_p) destiné à l'interprétation d'un script déterminé, de manière à remplir une tâche prédéterminée, en ce que ledit hyperlien définit l'adresse d'un premier agent intelligent traducteur de script supplémentaire (232a_p), et en ce que ladite seconde méthode comprend au moins les étapes subséquentes suivantes

consistant à décrire lesdits agents intelligents supplémentaires mis en œuvre pour accéder aux dits objets (*Obvi*) :

- a/ création d'une structure d'appel comprenant au moins un élément, un chaque élément comportant un identificateur d'un desdits agents intelligents supplémentaires (232a_p) et des données, dites argument, utilisées par celui-ci pour l'exécution de ladite tâche ;
- b/ l'établissement d'une session entre un premier agent intelligent supplémentaire (232a_p) et ledit agent intelligent (7) coopérant avec ledit système de gestion de fichier virtuel (8), cet agent intelligent supplémentaire (232a_p) étant sélectionné par l'identifiant d'un premier élément de ladite structure d'appel et retenant un élément de ladite structure qui lui est destiné ;
- c/ lorsque ladite structure comporte plusieurs éléments, établissement d'une session entre ledit premier agent intelligent supplémentaire (232a_m), et un autre agent intelligent supplémentaire (232a_n), et transmission des éléments restants de la structure d'appel, cet autre agent intelligent supplémentaire (232a_n) étant sélectionné par ledit identifiant et retenant un élément de ladite structure qui lui est destiné ; et
- d/ répétition de l'étape c/ jusqu'à épuisement des éléments constituant ladite structure d'appel.

13. Procédé selon la revendication 3, en ce que ledit navigateur "WEB" (10) transmet à ladite carte à puce (2*a*), dans une phase préliminaire, une première série de données de sécurité déterminant des droits d'accès aux dits objets (*Obvi*) et en ce que ladite liste des objets (*Obvi*) accessibles via ladite carte à puce (2*a*) est établie, sous la commande dudit agent intelligent traducteur de script (7) coopérant avec ledit système de gestion de fichier virtuel (8), par comparaison avec une seconde série de données de sécurité enregistrées dans une base de donnée de sécurité comprise dans ladite carte à puce (2*a*).

14. Procédé selon la revendication 13, caractérisé en ce que ladite première série de données de sécurité comprend un identifiant et un mot de passe.

15. Architecture pour la mise en œuvre du procédé selon la revendication 3, caractérisée en ce que :

- - ledit terminal (1) comprend au moins ledit navigateur "WEB" (10), ladite première pile protocolaire (101), et ladite première pièce de logiciel (13) constituant une couche protocolaire de communication spécifique, formant interface entre ces couches basses (101) et ledit navigateur "WEB" (10) ; et
- - ladite carte à puce (2a) comprend ladite deuxième pile protocolaire (200*a*), ladite seconde pièce de logiciel (23*a*) constituant une couche protocolaire de communication spécifique, ledit système de gestion de fichiers virtuel (8) et ladite deuxième entité logicielle constituée par un agent intelligent traducteur de script (7), cette dernière formant interface entre ledit système de gestion de fichiers virtuel (8) et ladite seconde pièce de logiciel (23*a*).

16. Architecture selon la revendication 15, caractérisée en ce que ledit système de gestion de fichiers virtuel est enregistré dans une mémoire re-programmable (8) comprise dans ladite carte à puce (2*a*).

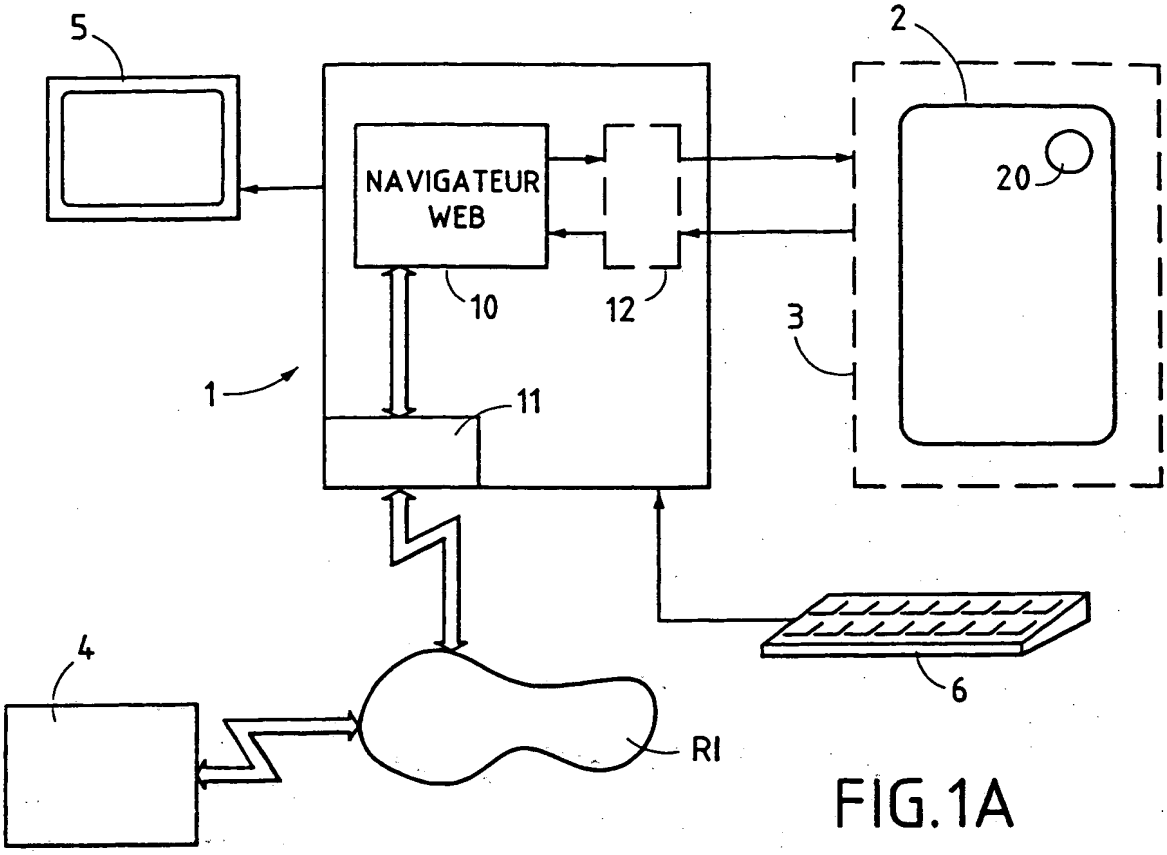

5
2
NAVIGATEUR WEB
20
10
12
3
1
11
4
RI
6

FIG.1A

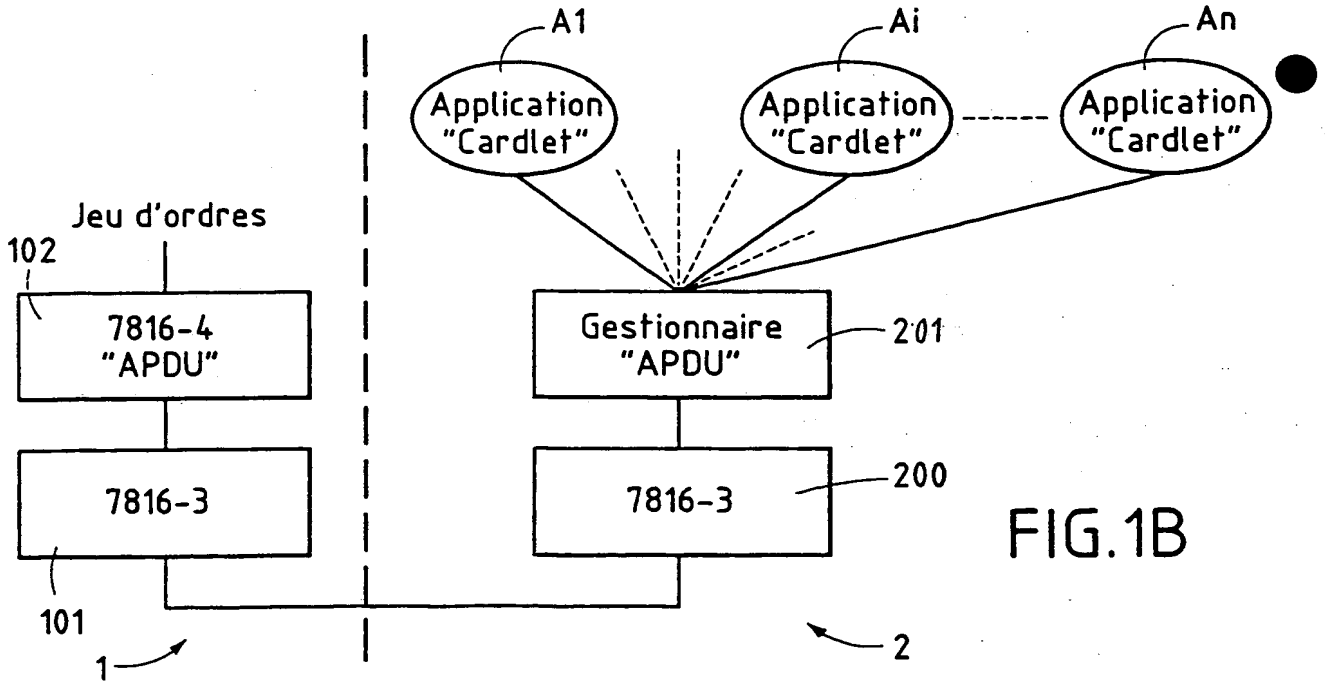

A1
Ai
An
Application "Cardlet"
Application "Cardlet"
Application "Cardlet"
Jeu d'ordres
102
7816-4 "APDU"
Gestionnaire "APDU"
201
7816-3
7816-3
200
101
1
2

FIG.1B

FIG.2

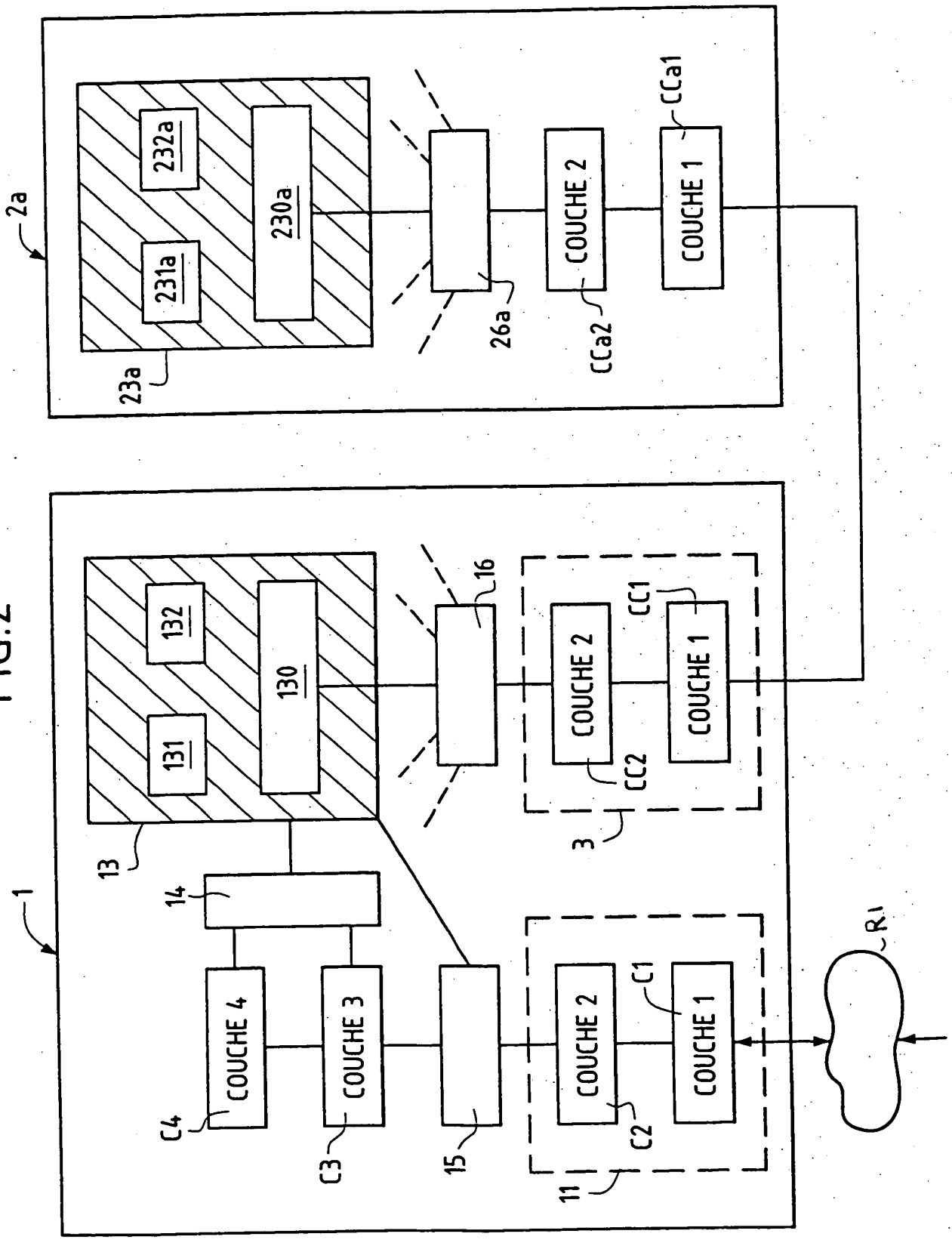
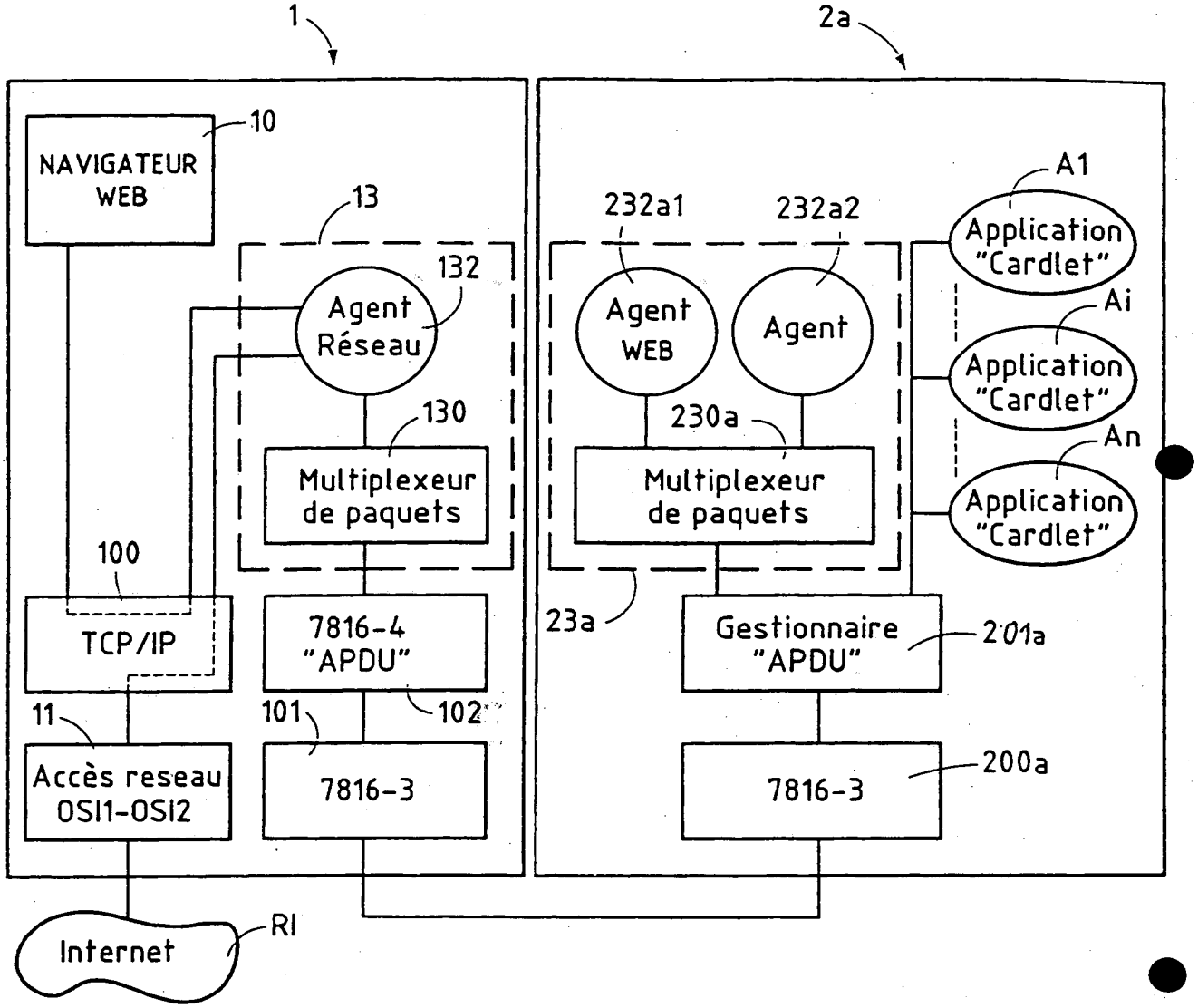

2a
232a
230a
231a
23a
26a
COUCHE 2
COUCHE 1
CCa2
CCa1
1
132
131
130
13
16
CC1
CC2
3
14
COUCHE 4
COUCHE 3
C4
C3
15
C1
C2
11
R1

FIG.3


1
2a
NAVIGATEUR WEB
10
13
132
Agent Réseau
130
Multiplexeur de paquets
100
TCP/IP
7816-4 "APDU"
11
101
102
Accès reseau OSI1-OSI2
7816-3
Internet
RI
232a1
232a2
A1
Application "Cardlet"
Ai
Application "Cardlet"
An
Application "Cardlet"
Agent WEB
Agent
230a
Multiplexeur de paquets
23a
Gestionnaire "APDU"
201a
7816-3
200a

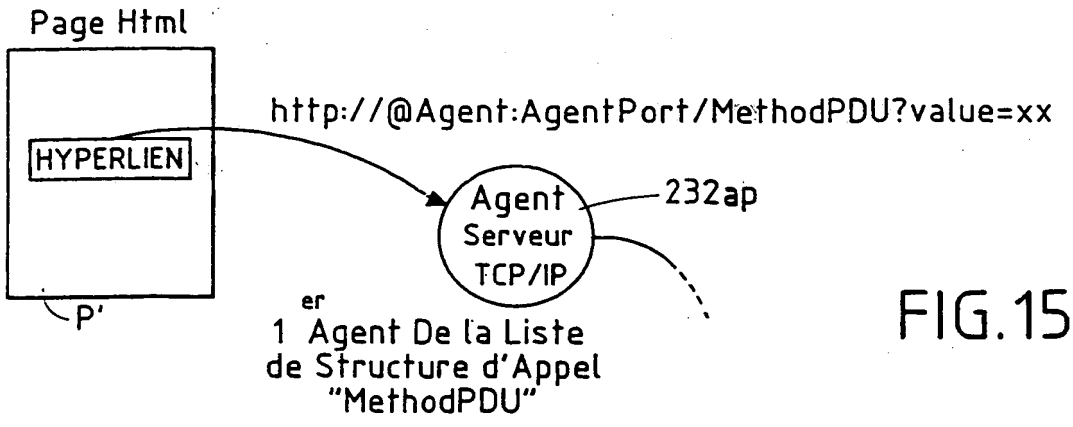

Page Html
HYPERLIEN
P'
http://@Agent:AgentPort/MethodPDU?value=xx
Agent Serveur TCP/IP
232ap
1er Agent De la Liste de Structure d'Appel "MethodPDU"

FIG.15

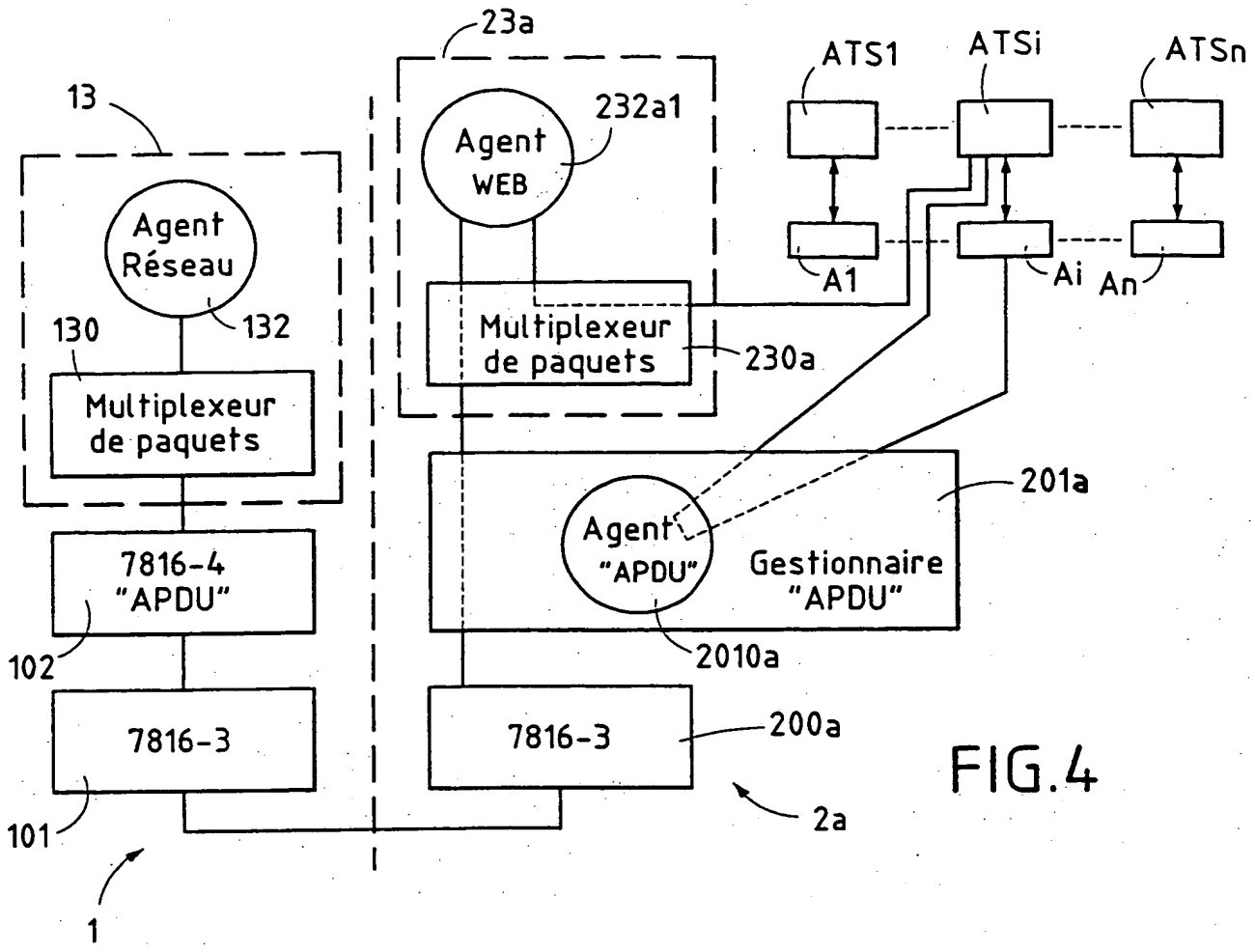

23a
ATS1
ATSi
ATSn
13
232a1
Agent
WEB
Agent
Réseau
130
132
Multiplexeur
de paquets
Multiplexeur
de paquets
230a
A1
Ai
An
201a
Agent
"APDU"
Gestionnaire
"APDU"
2010a
7816-4
"APDU"
102
7816-3
7816-3
200a
101
2a
1

FIG.4

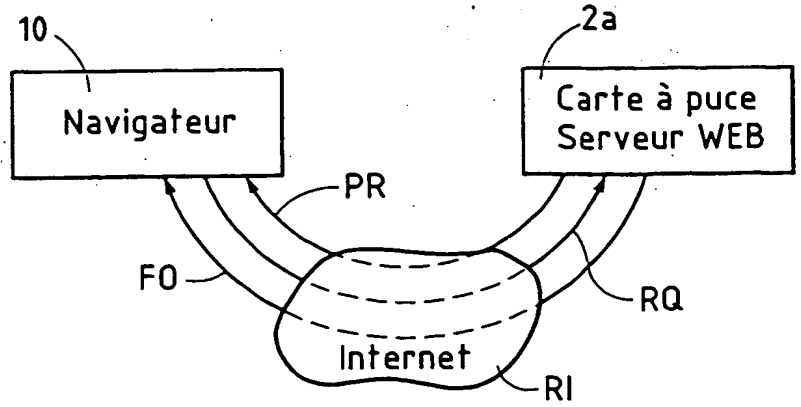

10
Navigateur
2a
Carte à puce
Serveur WEB
PR
FO
RQ
Internet
RI

FIG.5

FIG.6

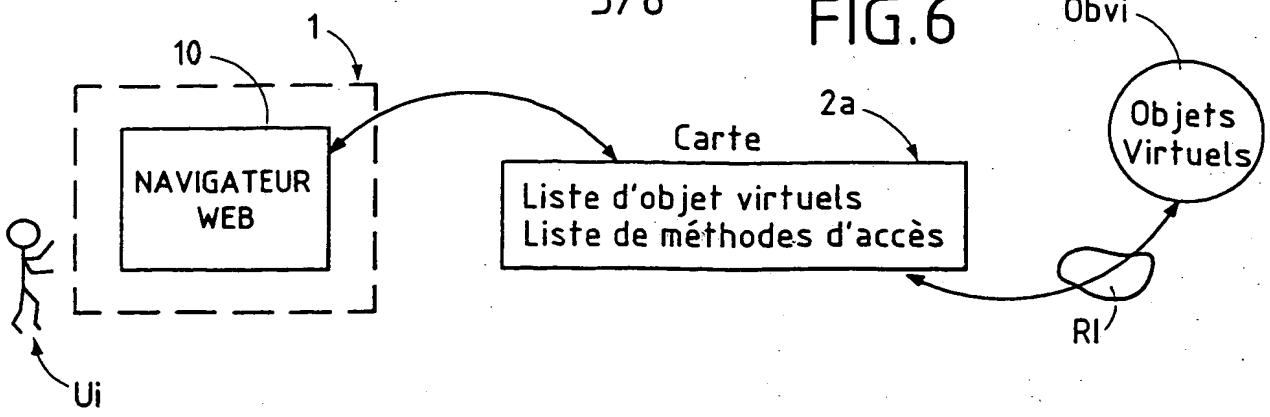
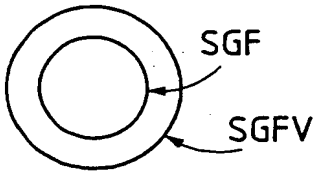


FIG.7



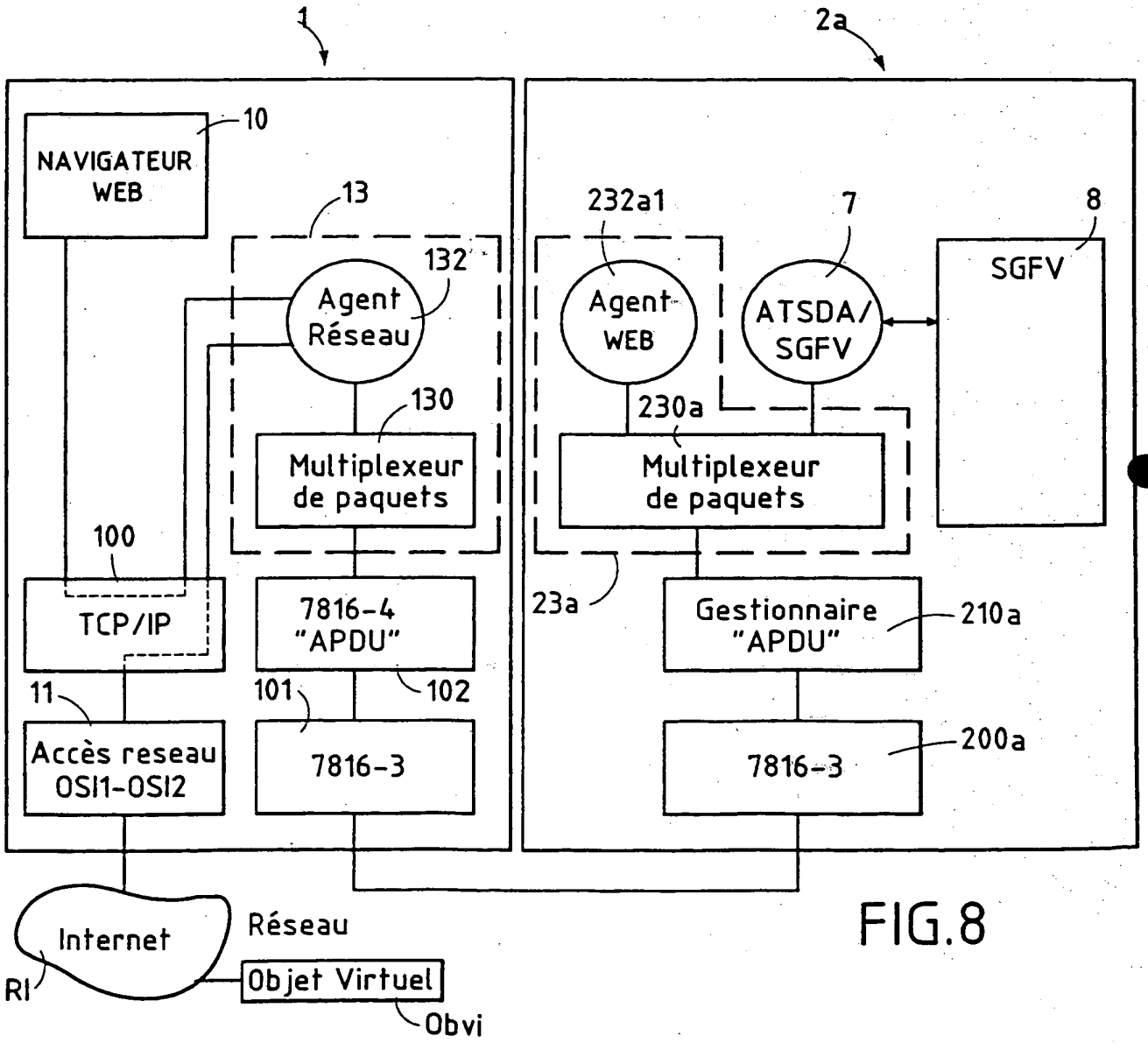


FIG.8

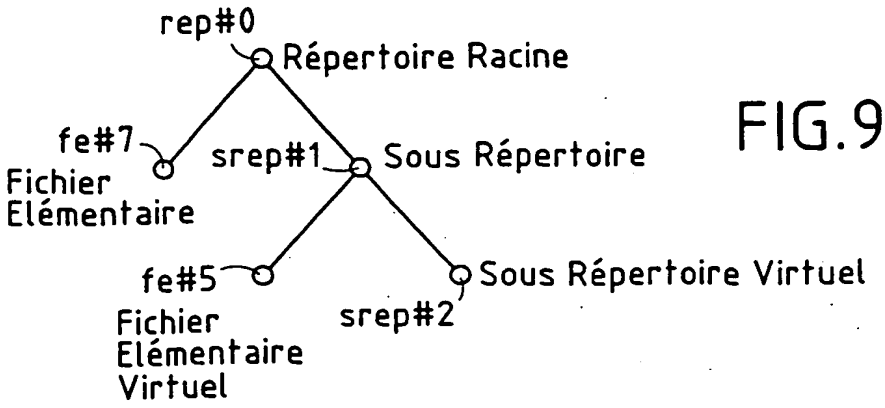


FIG.9

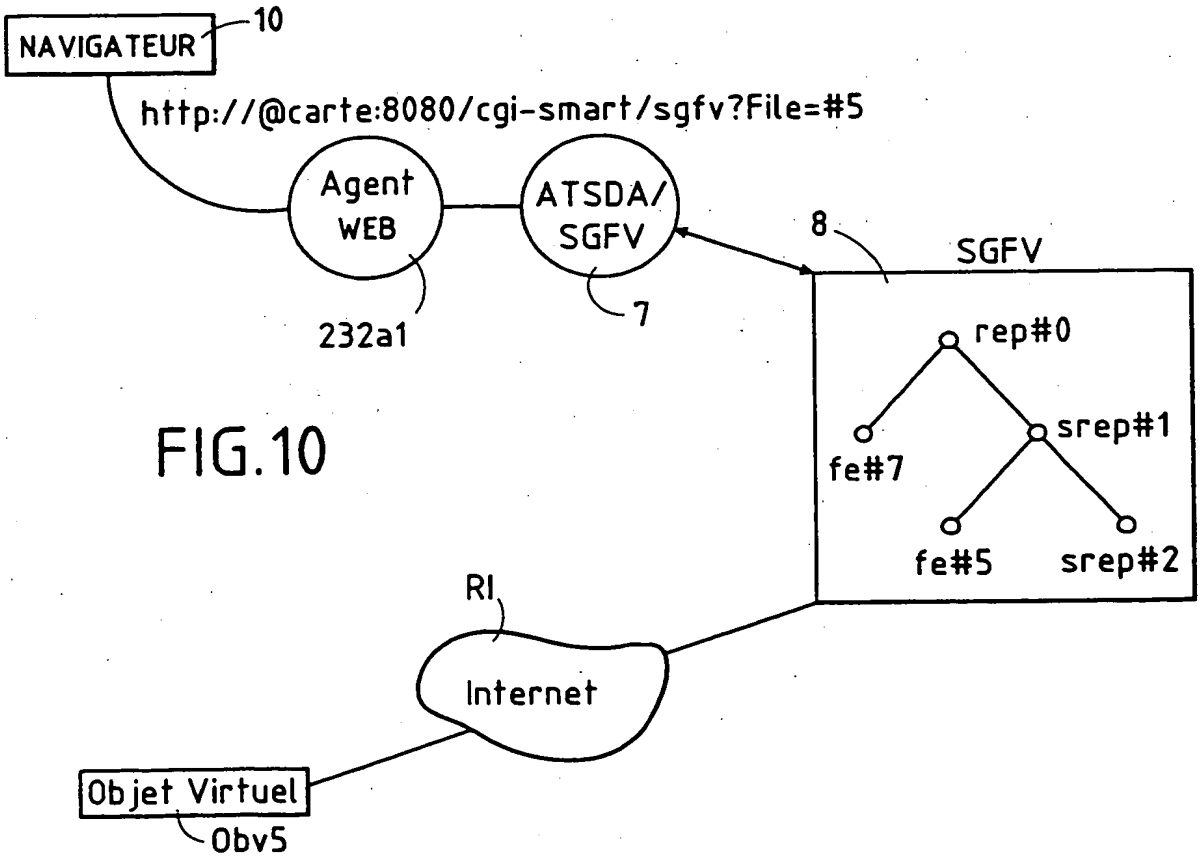


FIG.10

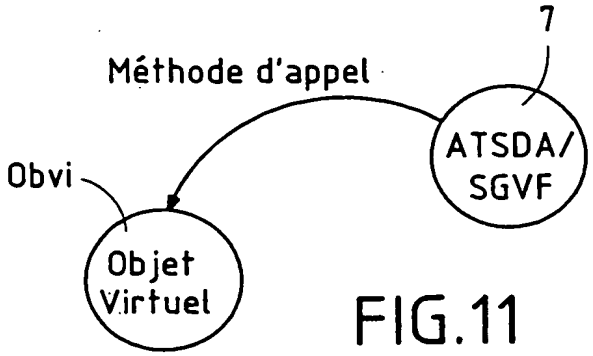


FIG.11

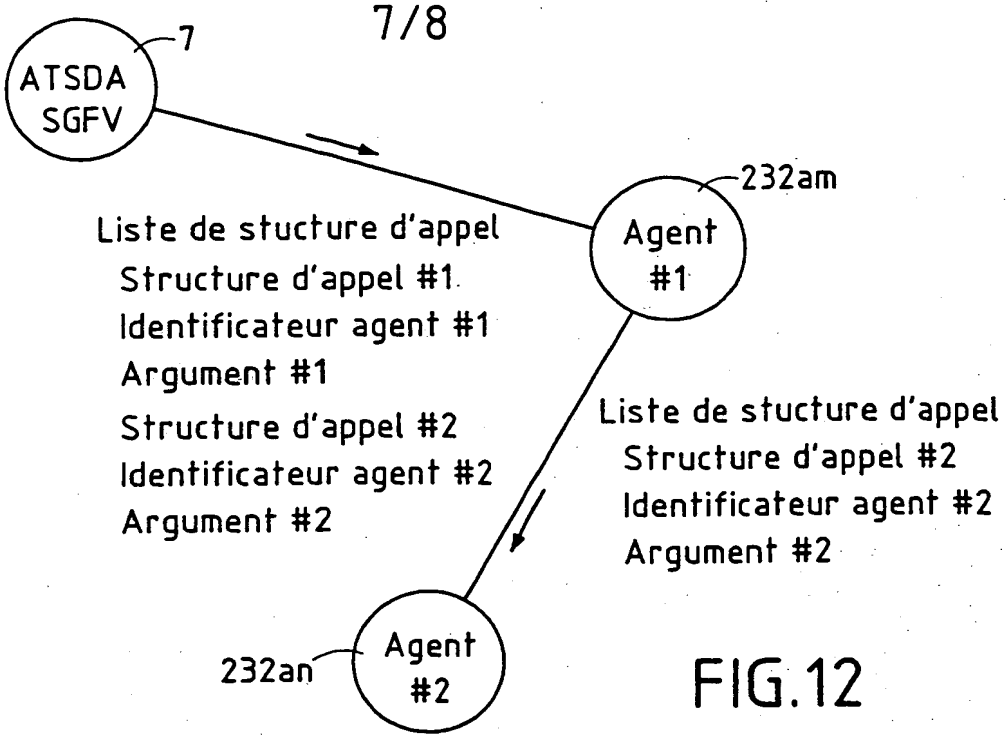

ATSDA
SGFV
7
232am
Agent
#1
Liste de stucture d'appel
Structure d'appel #1
Identificateur agent #1
Argument #1
Structure d'appel #2
Identificateur agent #2
Argument #2
Liste de stucture d'appel
Structure d'appel #2
Identificateur agent #2
Argument #2
232an
Agent
#2

FIG.12

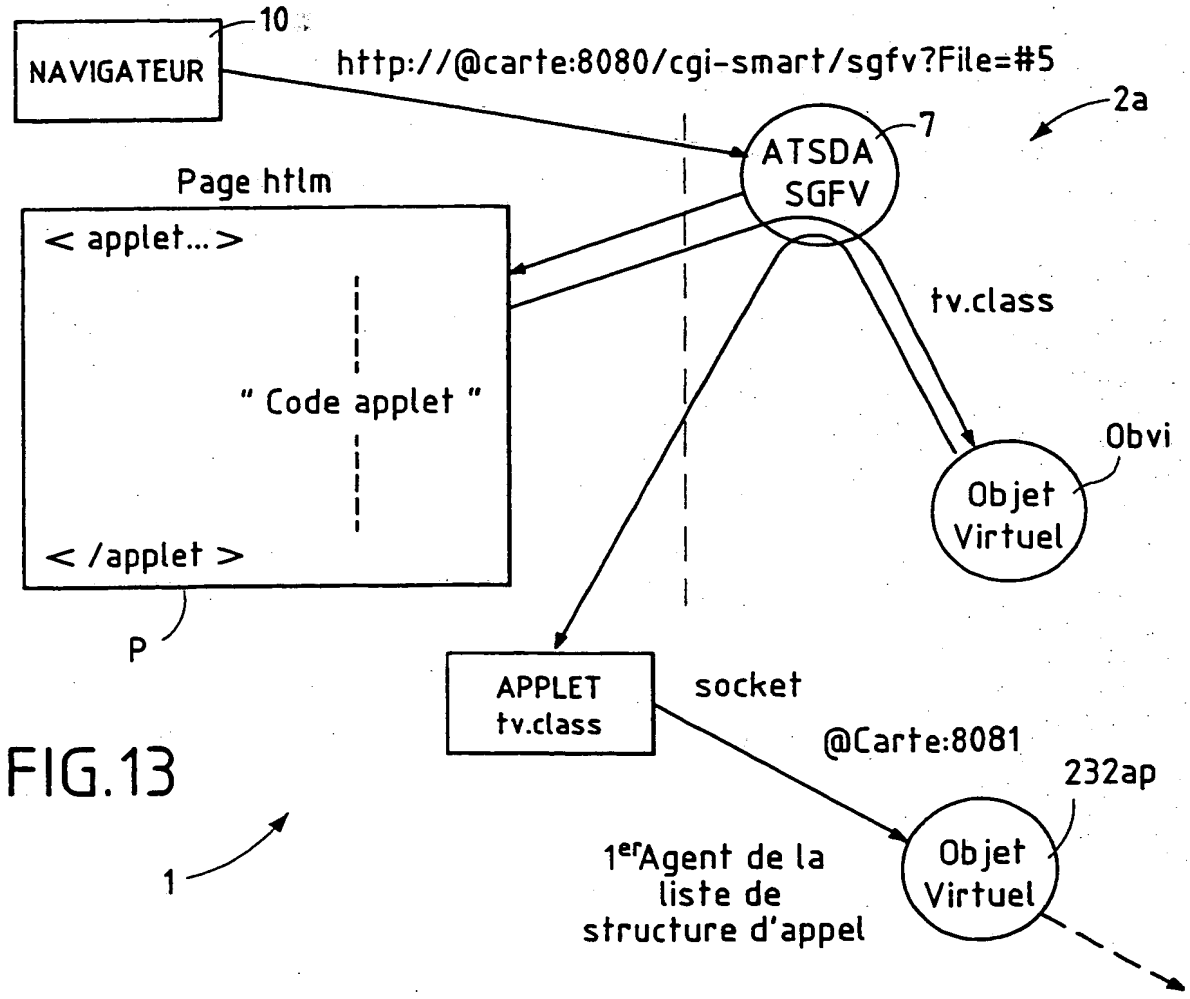

NAVIGATEUR
10
http://@carte:8080/cgi-smart/sgfv?File=#5
2a
ATSDA
SGFV
7
Page htlm
< applet... >
" Code applet "
< /applet >
P
tv.class
Obvi
Objet
Virtuel
APPLET
tv.class
socket
@Carte:8081
232ap
Objet
Virtuel
1er Agent de la
liste de
structure d'appel
1

FIG.13

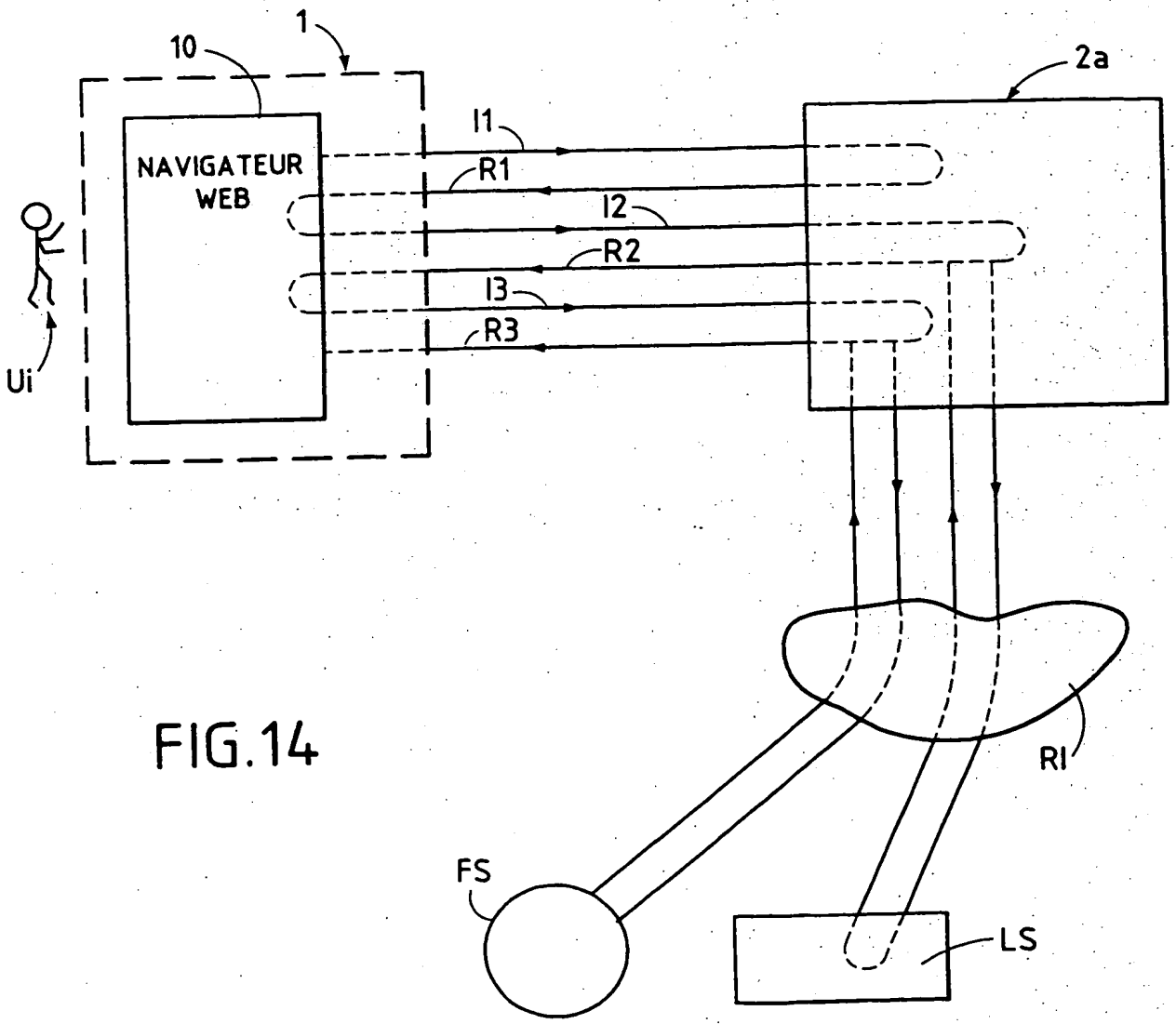


FIG.14

This Page Blank (uspto)